MANIPULACIÓ GEOMÈTRICA DELS BLOCS D'AUTOCAD

Carles Nogués Mañé (àlies Joan Colom),
professor del Departament d'Expressió Gràfica a l'Enginyeria (Universitat Politècnica de Catalunya)

Barcelona, desembre de 2007

ÍNDEX:

INTRODUCCIÓ

Si el text que teniu entre mans s'hagués d'adscriure a una categoria determinada, podríem parlar de recull de procediments per inserir els blocs d'AutoCAD de forma més eficient, en la resolució de problemes prèviament tipificats. Dir-ne llibre seria inexacte a més de pretenciós: mai no hi ha concorregut la voluntat de ser exhaustiu ni d'aconseguir certa homogeneïtat en el tractament d'aquests problemes, que només tenen en comú la matèria primera objecte de manipulació; pel que fa als mitjans, la PRIMERA PART descriu protocols d'actuació que l'usuari haurà d'aplicar manualment, mentre que la SEGONA PART ofereix rutines programades en AutoLISP i VisualLISP que l'eximiran d'aquesta obligació. Exposat així, ras i curt, podria semblar que els mateixos mètodes manuals presentats en primer lloc són després els que AutoLISP automatitza; per això convé aclarir d'entrada que la problemàtica de la PRIMERA PART, tot i que pròxima a la de la SEGONA, és diferent i reprodueix el contingut d'uns apunts que, amb el títol BLOCS I GEOMETRIA: 5 EXERCICIS COMENTATS, formen part del material de suport a l'assignatura ELEMENTS DE CAD impartida per l'autor en l'Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona. Aquesta monografia tenia per objecte cobrir el buit bibliogràfic que es detectava en el vessant geomètric de la inserció de blocs, a diferència del que s'interessa per l'estructura de dades més adient en cada context (incrustació de dibuixos amb ***INSERT*** *versus* vinculació mitjançant ***REFX***), més profusament tractada, tot proposant sistematitzar tipològicament els casos on l'escala és funció lineal de distàncies. Us en fareu millor el càrrec llegint-ne la introducció, que traslladem aquí:

*** *** ***

L'optimització de l'ús de blocs en AutoCAD, pel que fa a incorporar en el dibuix figures o elements gràfics (conjunts d'objectes, en definitiva) repetits, amb el doble criteri econòmic de minimitzar la intervenció de l'usuari (estalvi de feina) i la informació redundant (estalvi d'ocupació en RAM primer, i en disc, després), té dues vessants: l'algorísmica (com cal adaptar el plantejament geomètric a les possibilitats de les ordres ***BLOQUE*** i ***INSERT***) i la d'estructura de dades (si convé incrustar la definició de la figura-hoste en els dibuixos-amfitrió, amb ***INSERT***, o vincular-la, amb ***REFX***, segons el context). Com que la majoria de textos acostumen a dedicar algun apartat al segon tema però rarament inclouen exemples que deixin entreveure la complexitat que pot arribar a tenir el primer, aquestes pàgines se centraran en aquest aspecte, amb cinc casos representatius.

Per ser la forma gràfica que més sovint ens ha acompanyat mentre dibuixàvem amb AutoCAD, clara, senzilla i amb una localització intuitiva del punt base i de les dimensions **x** i **y**, la figura que protagonitza els exemples és una reproducció del símbol controlat per l'ordre ***SIMBSCP***, opció ***Origen***, referit a l'espai model i en visió no perspectiva, únic vigent fins el canvi produït en la versió ACAD 2002.

Sota el lema que no importa massa que la creació d'un bloc i demés preparatius encaminats a facilitar el conjunt d'insercions portin molta feina (al cap i a la fi, parlem d'operacions que es realitzaran un sol cop), sempre que això serveixi efectivament perquè en cadascuna d'aquestes insercions la intervenció de l'usuari sigui mínima, aquests casos revelaran maneres específiques d'abordar cinc dels sis aspectes crítics de la manipulació de blocs:

1) Suposant que ja disposéssim del bloc, com ens convindria determinar els factors d'escala (per exemple, entrant posicions **P** i **Q** representatives d'una magnitud proporcional a la grandària de la figura, considerada globalment o separant les direccions **X** i **Y**).
2) Suposant que ja disposéssim del bloc, com ens convindria determinar l'angle de rotació (per exemple, entrant posicions **R** i **S** representatives de l'orientació que haurà d'adoptar la figura).
3) Quin punt de la figura convé adoptar com a punt base o d'ancoratge del bloc (usarem indistintament aquests dos termes, lligats a l'ordre ***BLOQUE***, per diferenciar-lo conceptualment de cadascuna de les posicions concretes que li donem en el dibuix mitjançant l'ordre ***INSERT***, que anomenarem punts d'inserció), qüestió lligada a les dues precedents, és en general una de les decisions a prendre. En els casos que estudiem aquí, tanmateix, el punt base ja està implícit en l'enunciat: és el nus de la creueta que en el símbol ***SCP*** identifica l'origen de coordenades.

4) Quina grandària haurà de tenir, en les direccions **X** i **Y**, la figura que usarem com a bloc (tinguem present que, si **A** i **B** són els punts a situar en **P** i **Q**, la distància **A-B** cal que sigui **1**).
5) Quina orientació cal donar a la figura que usarem com a bloc (tinguem present que, si **C** i **D** són els punts a situar en **R** i **S**, respectivament, **D** haurà de quedar a la dreta de **C**, sobre la mateixa horitzontal, per tal que l'angle **C-D** sigui **0**).
6) En quin ordre cal subministrar a ***INSERT*** les dades geomètriques (punt d'inserció **I**, factors d'escala i angle de gir), en funció d'allò que haguem decidit a 1, 2 i 3, ja que l'ordre dependrà de si escala i angle han de ser ínputs numèrics o gràfics i, en el segon cas, de si el primer punt està o no implícit (**I = P**, si parlem de l'escala; **I = R**, si parlem de l'angle).

Tret de l'Exercici 1, on encara serà possible executar ***INSERT*** respectant "l'ordre natural" d'introducció de les dades, en la resta d'exemples caldrà forçar aquest ordre, posposant el punt d'inserció a una determinació dels factors d'escala que, paradoxalment, necessita recolzar-se en aquest punt. Així doncs, en els Exercicis 2 al 5 l'usuari no podrà decidir sobre la marxa (quan l'ordre ***INSERT*** li ho demani) on insereix el bloc sinó que, una de dues: o abans de començar a fer insercions dibuixa tants objectes punt (ordre ***PUNTO***) com blocs vulgui plantar, o ho va fent un a un, just abans de cada inserció (en aquest cas, en comptes de dibuixar punts que al final haurem d'esborrar, podem substituir ***PUNTO*** per l'orde ***ID***, que no deixa rastre però que un cop a ***INSERT*** permetrà fer referència a la posició assenyalada, escrivint **@**).

Abans de donar pas als exemples concrets, els situarem en el context d'una doble classificació dels conjunts de figures susceptibles de ser reproduïdes mitjançant la inserció d'un prototipus amb factors d'escala i angles de gir no arbitraris (geomètricament parlant), amb el propòsit que el lector tingui noció del variat repertori de problemes gràfics que poden trobar un tractament satisfactori en la utilització de blocs. Els qui estiguin interessats a plantejar-se nous exercicis, corresponents a categories de la classificació no contemplades pels cinc que aquí es presenten, faran bé de seguir llegint aquesta introducció. Qui només pretengui preparar-se amb aquests cinc exercicis per afrontar amb garanties el Segon Control de l'assignatura ELEMENTS DE CAD, pot saltar-s'ho i passar directament al primer de la sèrie. (Cal aclarir, però, que aquesta preparació passa necessàriament per realitzar els exercicis, seguint el discurs pas a pas, ja que conscientment s'ha defugit la preparació d'unes receptes de consulta immediata, que és just allò que voldrien tots aquells a qui preocupa molt més aprovar que aprendre: qui pensi que obrint aquestes pàgines per primer cop en el moment del Control en treurà alguna cosa, ho té clar...)

D'una banda, classificarem aquests conjunts segons que les mides **x** i **y** dels elements gràfics siguin funció lineal de la distància (des del punt d'inserció) a un punt o a una corba:

1- Les mides **x** i **y** varien linealment amb la distància a punts, poguent distingir els casos
   1.1- En què varien linealment amb la distància al mateix punt, depenent de
      1.1.1- Si la llei de variació és la mateixa (en aquest supòsit, tots els elements han de ser figures semblants geomètricament).
      1.1.2- Si les lleis de variació de **x** i de **y** són diferents.
   1.2- En què **x** varia linealment amb la distància a un punt i **y** varia linealment amb la distància a un altre.

2- Les mides **x** i **y** varien linealment amb la distància a corbes, i en aquest cas caldria definir la distància, precisant si es pren en una direcció fixa (per exemple, distància presa horitzontament, es a dir, fins al punt en què una recta paral·lela a l'eix **X**, traçada des del punt d'inserció de l'element, intercepti la corba), relativa a la corba (p. ex., normal a la corba) o relativa al mateix element inserit (p. ex., distància fins al punt en què una recta traçada des del punt d'inserció de l'element, en la direcció **X** pròpia de la seva definició de bloc, intercepti la corba), poguent distingir els casos
   2.1- En què varien linealment amb la distància a la mateixa corba, depenent de
      2.1.1- Si la llei de variació és la mateixa (en aquest supòsit, tots els elements han de ser figures semblants geomètricament).
      2.1.2- Si les lleis de variació de **x** i de **y** són diferents.

2.2- En què **x** varia linealment amb la distància a una corba i **y** varia linealment amb la distància a una altra.

3- De les mides **x** i **y**, unes varien linealment amb la distància a un punt i les altres varien linealment amb la distància a una corba.

De l'altra, els classificarem segons que els elements (girant al voltant del punt d'inserció) s'orientin en relació a un punt o a una corba:

A- Elements orientats cap un punt (caldria definir l'orientació: p. ex. girat 90° respecte la recta que uneix el punt d'inserció de l'element amb aquest punt).

B- Elements orientats cap una corba (caldria definir l'orientació: p. ex., que la recta que surt del punt d'inserció de l'element en la direcció **X** pròpia de la seva definició de bloc talli la corba ortogonalment). En aquesta categoria podem distingir els casos
   B.A- En què la corba és una recta, raó per la qual tots els elements tenen la mateixa orientació (absoluta).
   B.B- En què la corba no és una recta i, en conseqüència, l'orientació dels elements no és uniforme.

Dins d'aquesta doble classificació,

- L'Exercici 1 seria, dintre d'un supòsit 2.2/B.A, el cas particular en què les dues corbes que determinen les mides en **x** i en **y** són les rectes **x = 0** i **y = 0** (és a dir, els eixos coordenats **Y** i **X**), amb les distàncies perpendiculars a aquestes (les coordenades $\mathbf{I_x}$ i $\mathbf{I_y}$ del punt d'inserció), i en què l'orientació uniforme de **0°** es presenta com a paral·lelisme entre l'eix local **X** del bloc i la recta **y = 0** (eix coordenat **X**).

- L'Exercici 2 seria, dintre d'un supòsit 2.1.1/B.A, el cas particular en què la corba única és una recta, la distància es pot definir en qualsevol direcció (tot i que, com veurem, resulta més operatiu prendre-la normalment a aquesta recta) i, com a l'Exercici 1, l'orientació també és uniforme i de **0°**.

- Els Exercicis 3 i 4 respondrien al supòsit 1.1.1/A, amb una orientació com la que, a tall d'exemple, s'apunta en la categoria A, però d'un exercici a l'altre la llei de variació de les mides **x = y** canvia qualitativament.

- L'Exercici 5 respondria al supòsit 1.1.2/A, amb una orientació centrífuga igual a la dels dos exercicis precedents, però amb una llei de variació de les mides **x** anàloga a la de l'Exercici 4 i una llei de variació de les mides **y** anàloga a la de l'Exercici 3.

En tots cinc, el tema clau és el primer dels sis aspectes crítics enumerats abans (com determinar el factor o factors d'escala), i comporta saber passar de la llei explicitada per l'enunciat (la grandària **l** de l'element és funció lineal de la distància **d** a un punt o corba, **l = A d + B**) a una nova formulació d'aquesta llei que permeti adaptar-la a les regles de joc de la inserció de blocs (la grandària **l** de l'element és proporcional a la distància **d'** a un altre punt o corba, **l = A' d'**, d'on deduirem la mida del bloc, $\mathbf{l_u = A'}$ i el factor d'escala **E = d'**).

NOTA: Les transcripcions del diàleg entre l'Editor de Dibuix d'AutoCAD i l'usuari (textos en *cursiva* i ***negreta cursiva***, respectivament) corresponen al producte ACAD 2000 (versió 15 d'AutoCAD) en castellà.

*** *** ***

Si no s'hagués advertit prèviament de la procedència d'aquesta introducció, ben segur que la lectura de la frase entre parèntesis precedint la doble classificació l'hauria delatada com a pròleg d'algun text docent. I no només per la referència explícita a l'assignatura i als controls establerts, sinó per l'ènfasi amb què s'insta l'estudiant a seguir l'argumentació en comptes d'anar de dret cap a unes conclusions que més d'un (millor no aventurar percentatges) voldria resumides en fitxes de consum immediat, per no haver d'entrar en disquisicions de mal pair.

A la SEGONA PART tampoc no s'ha anat al gra (del seu embalum en pot donar idea una ràpida ullada a l'ÍNDEX), si per tal cosa s'entén limitar-se a reproduir el codi font de les rutines a què feiem esment en el primer paràgraf. Tanmateix, abans de continuar per justificar-ne la part discursiva, donarem satisfacció als qui només s'interessin pel seu valor instrumental (al cap i a la fi és una opció legítima), amb la localització precisa d'aquest codi:

- Entre les pàgines 75 i 81 (OPTIMITZACIÓ DELS RECURSOS EXISTENTS: 2 EXEMPLES) hi trobareu el codi relatiu a les noves ordres ***GINSERT*** (*INSERT Girat*) i ***RATREDIT*** (*ATREDIT Repetitiu*).

- Entre les pàgines 273 i 274 (ENCAIX 3D DE BLOCS AMB ATRIBUTS SITUATS LLIUREMENT, tot i que la definitiva VERSIÓ 1++ que s'hi ofereix es basa en una versió prèvia desenvolupada en el capítol NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS) hi trobareu el codi relatiu a la nova ordre ***INSERTOK*** (alternativa a *INSERT* per tal d'encaixar un bloc 2D sense atributs en un paral·lelogram qualsevol).

- Entre les pàgines 290 i 304, 320 i 323, i 325 i 326 (ENCAIX 3D... : OPTIMITZACIÓ DE L'ESTRUCTURA DE BLOCS) s'ofereix el plat fort d'aquest treball: el programa que permet l'encaix d'un bloc 2D o 3D, amb atributs (eventualment situats fora del pla ***XY*** del seu sistema de referència) o sense, en qualsevol paral·lelogram o paral·lelepípede, mitjançant la incorporació de les noves ordres ***INS2D*** i ***INS3D*** (principals), i ***BLOQUEOK*** i ***DESCOMPOK*** (auxiliars). De les tres localitzacions, a la primera hi tenim la VERSIÓ 22++ completa, a la segona hi figuren les esmenes que donen lloc a la definitiva VERSIÓ 23++ i la tercera posa fi a aquesta SEGONA PART amb una breu guia que explica l'actuació de cadascuna de les quatre ordres.

- Entre les pàgines 343 i 348 (annex ENFOC ALGEBRAIC DE LES TRANSFORMACIONS AFINS) hi teniu un grapat de funcions (algunes reciclades d'altres treballs de l'autor) al servei de les noves ordres ***DEF-TRANSF***, ***APL-TRANSF-V*** i ***APL-TRANSF-N*** que, tot i que situades fora de l'orbita dels blocs d'AutoCAD, s'inscriuen en l'estratègia comuna de definir transformacions afins 3D a partir de les condicions d'encaix.

Com que hem parlat de "plat fort", no voldríem deixar els addictes al menjar ràpid amb l'ai al cor i, per si això de l'encaix en paral·lelograms o paral·lelepípedes no els ha donat cap pista, tot i ser prematur farem un avanç d'entrada en matèria. Si s'ha tingut la precaució de referir un bloc 2D a un quadrat unitari ortogonal, inserir-lo de manera que s'adapti a qualsevol marc rectangular establert en el dibuix serà immediat: de fet, l'EXERCICI 1 de la PRIMERA PART seria una variació sobre aquest tema, considerant que no hi ha gir (rectangle d'encaix ortogonal), i la nova ordre ***GINSERT*** del capítol introductori a la SEGONA PART amplia les opcions de ***INSERT*** perquè l'operació esmentada també sigui possible quan n'hi ha (rectangle girat). Tanmateix, la immediatesa desapareix si pretenem encadenar insercions de manera que, a més d'una combinació simple de escalat, gir i translació, l'acció dugui implícita una transformació de cisallament. Perquè és clar que si inserim el bloc girat i convertim la inserció en un bloc que al seu torn tornem a inserir, ara però amb escalat no uniforme, la figura transformada del quadrat de referència primitiu serà un paral·lelogram, però el problema és: dibuixat un marc romboïdal concret, ¿amb quin gir caldrà realitzar la primera inserció, i quin gir i factors d'escala haurem d'aplicar en la segona perquè el quadrat de referència s'adapti al marc? El problema encara es complica més si volem aprofitar el resultat de la primera inserció per a encaixos en d'altres paral·lelograms, creant un sistema no redundant de insercions intermèdies. Doncs bé: ***INS2D*** i ***INS3D*** resolen tot això en 2D i 3D, i són aplicables a blocs proveïts d'atributs, no només dels continguts en el pla de base (que són els únics que l'ordre ***INSERT*** sap situar correctament) sinó també dels ubicats i orientats lliurement.

Havent satisfet ja aquesta necessitat peremptòria, amb la menció concreta a unes ordres que amplien el repertori d'AutoCAD pel que fa als blocs, ara ens permetrem d'ocupar-nos de qüestions més generals, i ho farem tot responent tres preguntes: ¿per què en aquesta SEGONA PART, en què no podem al·legar la procedència didàctica del material, també ens esplaiem amb tota mena d'explicacions, com a la PRIMERA?; ¿per què l'autor s'ha dedicat a aquestes activitats, si ningú no li ho demanava?; ¿per què juga a desinteressat i solidari, donant lliure difusió a un *software* que precisa de la plataforma AutoCAD, comercialitzada per una multinacional del ram? O, més abruptament encara: ¿no és això, més aviat, promoció encoberta?

La resposta a la primera qüestió és doble. D'una banda hi ha la necessitat que té l'autor de deixar constància escrita de tot allò que va fent, de qualsevol manera i a qualsevol paper (sobretot quan cal dedicar-s'hi a estones perdudes i, sovint, aparcar-ho per llargues temporades) i, de tant en tant, de reunir i posar en ordre les notes disperses (si més no, d'anar guardant les successives versions del codi) per poder retrobar el fil del discurs quan hi hagi l'ocasió. D'altra banda, de la necessitat n'ha volgut fer virtut pensant en la possibilitat remota d'interessar algú més enllà de la vessant utilitària del producte. Així, a fi de permetre que aquells prou motivats (i amb més paciència que Job, tot s'ha de dir) segueixin les passes de l'autor i puguin treure'n l'entrellat per esmenar errors, per completar carències, per millorar l'eficiència dels algorismes o per anar més enllà, aquest modest servidor encara ha farcit amb més explicacions enunciats que tal vegada ja eren prou evidents per al lector mitjà. Si admetem que aquestes explicacions, més cops del que fóra recomanable, responen a problemes que se li van presentar en el transcurs d'una ascensió accidentada i sinuosa però que, des de la perspectiva del cim ja assolit, perden sentit i fins i tot distreuen del rumb correcte, no seria estrany que a la pràctica només servissin per dissuadir alguns lectors potencials.

Pel que fa a les motivacions, si cal pronunciar-se sobre les qüestions de principi l'autor es limitarà a manifestar el seu desacord amb la desimboltura amb què les nostres autoritats acadèmiques s'omplen la boca amb la paraula excel·l·l·l·lència quan es refereixen a la recerca, com si l'excel·lència depengués d'un simple acte de voluntat, en comptes d'acceptar amb realisme que és un concepte estadístic, la quinta essència d'una producció extensa i sostinguda en el temps. Ni s'improvisa ni s'instaura per Reial Decret: únicament d'un brou de cultiu favorable i d'un fotimer de modestes iniciatives (les unes de reixides, les altres no, però cap ni una de menystenible) en podrà acabar florint l'excel·lència. Si tot el personal docent a temps complet (almenys el que imparteix assignatures que han superat el període fundacional o els traumàtics cops de timó per adaptar-se als nous plans d'estudis, que és la majoria) dediqués a la universitat 37 ½ hores per setmana, al llarg de 45 setmanes l'any (que per a això, ni més ni menys, paga el contribuent), la producció en I+D seria espectacular, i d'aquesta n'excel·liria probablement el mateix percentatge que ara, és a dir, més en termes absoluts. Però bàsicament hi ha dos obstacles que s'interposen entre aquesta ucronia i allò que tots coneixem. El primer és una pràctica que, no per generalitzada a tota la Universitat Pública (en la Privada almenys no constitueix un frau al contribuent), s'ha de silenciar. Són les que podríem anomenar "Activitats Externes Remunerades d'Alta Qualificació Professional" (massa llarg i de sigles poc eufòniques, però és que l'autor sempre s'ha resistit a utilitzar el carrincló eufemisme manllevat sense pudor de l'argot empresarial en anglès, que s'abrevia com Tele Taxi), que sovint han derivat cap a una modalitat peculiar de pluriocupació pluriretribuïda i que, de ser discretament tolerades han passat a ser recomanades com a exemple a imitar, prèvia l'adquisició d'una butlla que transmuta el vici en virtud per un mòdic 14,7%: si de debò el que s'embutxaquen aquests espavilats, que cobren dues hores treballant-ne una, tingués un caràcter d'incentiu, ¿no us semblaria més lògic invertir els termes per tal que la butlla representés el 85,3% i que el 14,7% residual fos precisament l'incentiu? El segon és l'escassetat real d'oportunitats de fer tasques de recerca que siguin reconegudes com a tals, atès l'esquizofrènic desdoblament dels criteris que aplica l'autoritat acadèmica a l'hora d'avaluar la I+D (excel·lència plasmada en articles publicats en revistes indexades o en patents guardonades), en franca contraposició a l'esmentada manca d'escrúpols en homologar tota mena d'activitats paral·leles (sempre que deixin calerons, és clar). A poques situacions els escau tan bé com a aquesta aquell refrany castellà que sentencia: *lo mejor es enemigo de lo bueno*. La trista conclusió és que, qui no està integrat en un grup de recerca amb rodatge i contactes ni té cap possibilitat de publicar en revistes indexades, intenta fer-se un sobresou sota el paraigua de la universitat (els més fervents defensors del lliure mercat solen practicar la competència deslleial, si en tenen ocasió) o es limita a cumplir les seves obligacions docents el millor que pot (que ja és molt). Tota aquest llarg prolegomen només era per situar al lector en el dilema en què es trobava l'autor, al qual va trobar una sortida segurament ingènua i estèril des del punt de vista de transformació de la realitat però perfectament legítima com a opció ètica individual, escapisme si es vol. Sense ser especialment masoquista, estava disposat a renunciar al reconeixement per la feina feta pero no al seu dret a fer recerca paral·lelament a la docència. Com que tampoc li venia de gust perdre temps trucant portes i dedicant-se a les relacions públiques, va decidir de tirar pel dret: si la jerarquia acadèmica tenia la facultat de decidir què era I+D i què no ho era, no seria ell qui en discutís l'autoritat; definiria lliurement els seus objectius, es posaria a pencar i d'això en diria J+E.

D'alguna manera es tractava de repetir les vivències personals de la Tesi Doctoral sense els condicionaments (cerca de l'estabilitat laboral, anar contra rellotge) que van desvirtuar allò que gairebé hauria pogut ser una experiència iniciàtica: programant era fàcil sentir-se el demiürg, en la mesura que un mateix es marcava les fites i només necessitava imaginació, prou de temps i ganes d'aprofitar-lo. A més la valoració dels resultats no depenia del parer d'altri, perquè l'ordinador era implacable però sempre et deia la veritat: allò funcionava o no funcionava. Potser és frívol reivindicar la faceta lúdica del treball, però considerava que el rendiment també està influït per les preferències personals i per la veterania que s'ha anat adquirint en un determinat domini, i que el fruit d'aquests jocs eren petites contribucions que algun dia podien veure la llum. Passava però que, quan ja havia aconseguit de resoldre el problema amb què ell mateix s'havia reptat, el tema perdia automàticament atractiu, li feia una certa mandra vestir el treball i no diguem vendre'l (això que ara en diuen "postproducció"), atesa sobretot la poca receptivitat de les forces vives departamentals. Però amb el pas dels anys, un dia un s'adona que ja fa temps que ha creuat l'equador, que el futur no és il·limitat i que potser ja va sent hora de recollir materials dispersos i posar-los en ordre. I ja en fa uns quants que l'autor reflexiona sobre la conveniència d'acabar coses deixades a mitges, posar-ne en solfa d'altres per tal de fer-les intel·ligibles i revisar algun treball que havia donat per bo però que, un cop deixat reposar, s'ha evidenciat coix, estrafet i clarament millorable, tot i mantenir la validesa dels pressupostos bàsics. Un d'aquests és l'aplicació AutoLISP ***INS2D***/***INS3D***, al voltant de la qual orbita la SEGONA PART.

Tot i que l'autor arrossegava feia anys la idea d'aplicar la construcció de Ritz (un mètode gràfic per trobar els eixos principals d'una el·lipse i la longitud dels diàmetres corresponents, a partir d'un parell qualsevol de conjugats) a la inserció de blocs, en la línia apuntada més amunt, no va ser fins a setembre de 1996 que s'hi va posar. El que ja no sabria explicar és com, havent-hi pel mig les classes a l'ETSETB i a l'EUPBL (que encara trigaria uns anys a desplaçar-se al Campus de Castelldefels, i ocupava a precari un edifici cedit per l'Ajuntament de Sant Just Desvern), a finals de gener de 1997 ja havia obtingut uns resultats i havia dut en persona a Autodesk Spain (llavors radicada al carrer Constitució 1, del mateix municipi, davant de l'EUPBL però a l'altra banda de l'autopista A-2) una carta, que transcriu tot seguit:

*** *** ***

AUTODESK, S.A. - SPAIN
C/ Constitució, 1
08960 SANT JUST DESVERN (Barcelona)

Me dirijo a Vds. para ofrecerles una rutina AutoLISP que puede ser de su interés. Entiendo que su generalidad justificaría incluirla en el paquete básico AutoCAD, conectándola en ACADR13.LSP al dispositivo AUTOLOAD: no se trata de una aplicación de carácter monográfico u orientada a una determinada actividad profesional; por el contrario, su propósito es complementar la orden INSERT, ampliando las posibilidades de uso de bloques en 2D y 3D.

Como profesor de CAD en la Escuela Técnica Superior de Ingenieros de Telecomunicación de Barcelona (Universidad Politécnica de Catalunya) y en la línea de desarrollo de aplicaciones informáticas del Departamento de Expresión Gráfica en la Ingeniería, desde hace algunos años consideraba la posibilidad de flexibilizar el uso de los bloques AutoCAD, mejorando la mecánica de su inserción. Aprovechando el pasado cuatrimestre un período de relativa calma, me planteé el problema y creo haberlo resuelto.

Sólo una incógnita se mantiene en el aire: ¿tratándose de algo a mi juicio tan elemental, es posible que nadie más haya tenido esta iniciativa? No me consta tal evento, aún admitiendo no haber perdido mucho tiempo en indagaciones porque el tema tampoco me preocupaba demasiado: personalmente me apetecía abordar la cuestión y llegar a unos resultados; ya los tengo y en principio eso me basta. Si además se da la circunstancia de que éste era un terreno inexplorado y que la nueva herramienta satisface una demanda real o que a juicio de los expertos la incorporación de la rutina contribuye a reforzar la imagen de AutoCAD como el más

completo de los sistemas de propósito general para PC, tanto mejor. Nadie en mejor posición que Vdes., que en definitiva son quienes deben pronunciarse sobre el particular, para opinar sobre ello con fundamento.

En el supuesto de estar en principio interesados, les ruego contacten conmigo antes de 60 días. Ya me indicarán si en lo sucesivo debo dirigirme a sus oficinas en Sant Just Desvern, Barcelona (España); Neuchatel (Switzerland) o San Rafael, California (USA). También dejo a su criterio establecer el protocolo de negociación. Entretanto, como es lógico, me reservo el acceso al código fuente. Naturalmente, ello no será obstáculo para realizar cuantas demostraciones estimen pertinentes, sobre dibujos y bloques facilitados por Vds. si así lo prefieren. Como carta de presentación, en el documento adjunto hallarán la descripción provisional del producto, suficiente a mi entender para hacerse una idea de sus características.

Por mi parte, y en el caso de prosperar la propuesta, sólo dos condiciones:

1) Deseo en todo momento mantenerme en el marco normativo establecido por la Universidad Politécnica de Catalunya para convenios con empresas, por lo que la prestación de servicios deberá tramitarse a través del Centro de Transferencia de Tecnología, que representa a dicha institución.

2) Con independencia de la cuantía finalmente establecida, la liquidación de honorarios deberá efectuarse en especie, entendiendo por tal el otorgamiento de licencias de software *Autodesk (de uso exclusivo para la enseñanza) al Departamento de Expresión Gráfica en la Ingeniería, cuya composición y número vendrán determinados por las necesidades de dicho departamento y el precio al que se comercialicen esas licencias en el momento de cerrar el acuerdo.*

Atentamente:

Carles Nogués Mañé *Tel. (93) 401 68 31* *EAN: nogues@ege.upc.es*
Departamento de Expresión Gráfica en la Ingeniería (UPC)
Edificio E.T.S. de Ingenieros Industriales de Barcelona, 5ª planta
Avda. Diagonal, 647
08028 BARCELONA

Barcelona, 28 de enero de 1997.

*** *** ***

Aquesta carta, a la qual s'havia adjuntat el text explicatiu *Perfil de la rutina INSERTOK.LSP* que no es reprodueix i un dibuix on l'escut constitucional d'Espanya, capturat d'alguna *web* estatal, s'inseria de **7 × 7** maneres diferents, no va tenir resposta. Des de la perspectiva actual això no l'estranya massa, i la temptativa la troba agosarada però ingènua. Però el que encara l'intriga és com va ser capaç d'ultimar en menys de cinc mesos unes versions en 2D i 3D que, malgrat les moltes insuficiències detectades anys després (amb l'excusa dels 60 dies, aquesta feina va quedar aparcada i el tastaolletes de l'autor va passar-se a un altre tema), funcionaven. Potser la raó és que el quadrimestre anterior s'havia hagut de fer càrrec de tres grups (dos de l'ETSETB i un de l'EUPBL), amb un total de 12 hores de classe a la setmana, i que el retorn a la dedicació normal de 8 hores, amb la possibilitat de realitzar paral·lelament tasques més creatives, l'havia estimulat i estava en plena eufòria. L'altra lectura, més negativa, és que en el decurs dels onze anys transcorreguts molts milions de neurones han quedat fora de servei...

En relació a les sospites de connivència amb Autodesk, Inc. que poguessin sorgir, l'autor ha de manifestar que mai no hi ha mantingut relacions comercials, i que els únics contactes que recorda amb Autodesk Spain (l'ofertiment sense resposta que acabem de referir i una consulta infructuosa a mitjans del 2004, que va incloure una petita picabaralla via e-mail) no van ser especialment efusius. Malgrat això, podria penjar-se la medalla d'haver estat pioner pel que fa a la incorporació del CAD a l'ensenyament, dintre dels estudis reglats de la UPC, amb els seus companys de departament Juan A. Soler i Lluïsa Palou. És més: fa uns deu anys hi havia una demanda tan insistent de cursos d'AutoCAD, que molts en el Departament d'Expressió

Gràfica a l'Enginyeria es feien creus que aquests professors assignats a l'ETSETB no s'animessin a muntar el *xiringuito* i n'organitzessin cursos adaptant mínimament l'experiència i el material didàctic disponible, decisió que a més de beneficis curriculars els n'hauria reportat de materials (prèvia liquidació del preceptiu 14,7% al CTT), però per raons diferents tots van coincidir a ignorar aquests cants de sirena. Després de la panfletada d'abans no cal dir quines eren les de l'autor.

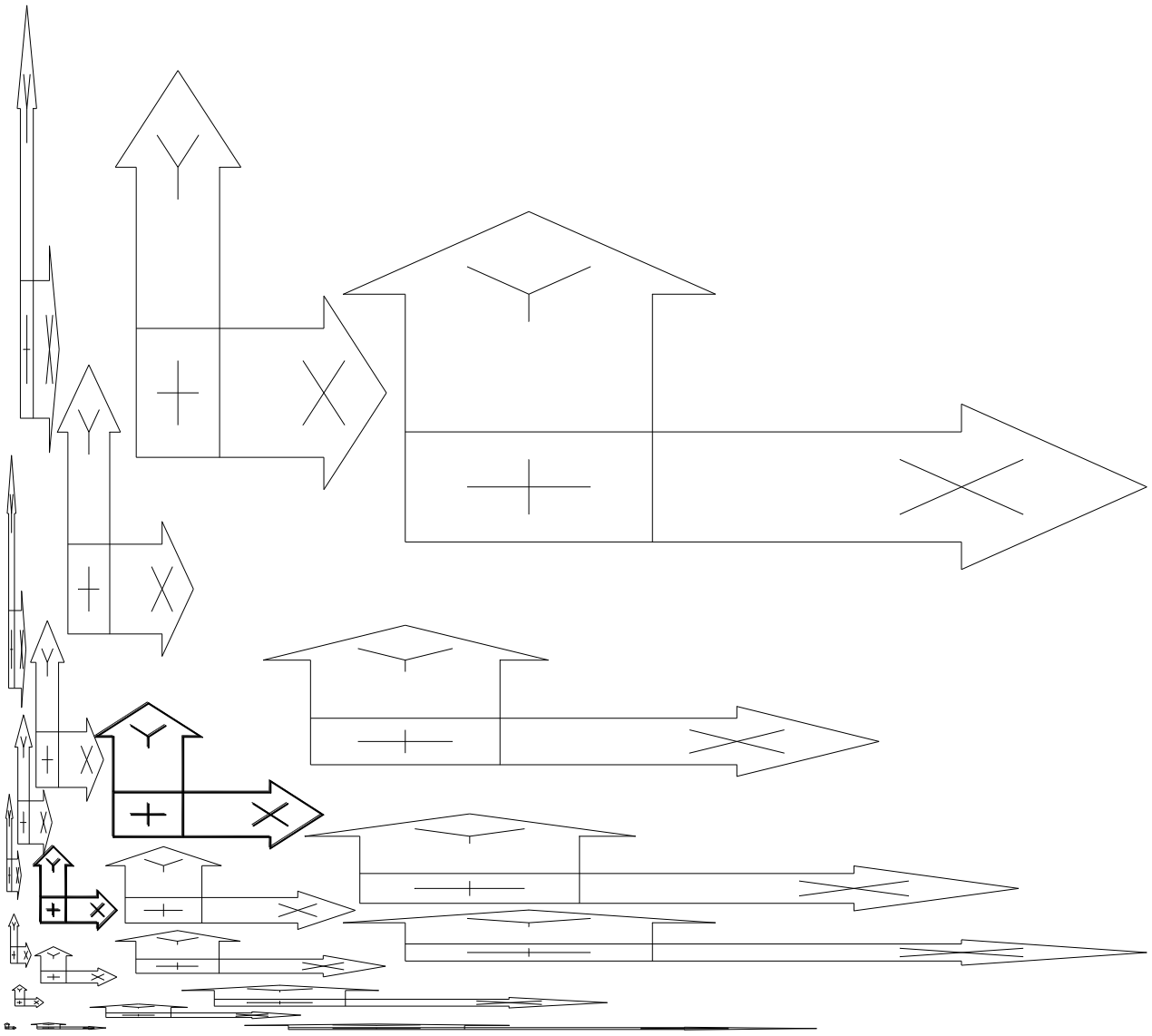
D'altra banda, aquest confessa que el motiva més programar que ensenyar geometria (que Sant Gaspard Monge el perdoni!) i, malgrat això, cada cop és més escèptic en relació al paper de les TIC en el desenvolupament de la Ciència, la Tècnica i el seu ensenyament. Ningú no nega l'impuls gegantí que aquesta simbiosi ha suposat, però tampoc a nigú no pot passar desapercebut com, a cavall de la dependència del *hard* i *software* que fan possible l'embranzida, l'escora cap a la privatització del coneixement sembla imparable. Exercir funcions docents a l'Universitat no és massa gratificant quan un s'adona que, en lloc d'intervenir en la transmissió d'un saber que és patrimoni col·lectiu i, per tant, difícilment alienable, està creant futurs usuaris d'un producte comercial, condicionant-los no només com a consumidors, que ja seria greu, sinó com a professionals, i tot això des de la Universitat Pública.

En aquest sentit, l'autor subscriu l'aposta estratègica de la UPC pel programari lliure fa més de cinc anys, tot i lamentar que amb la perspectiva del temps passat els resultats siguin més que modestos: com sempre, les iniciatives propiciades des de successius equips de govern han estat tímides i en alguns casos s'han estrellat contra la inèrcia departamental, alimentada a parts iguals per interessos creats i per la resistència atàvica al canvi. Val a dir que l'autor tampoc no és immune al conservadorisme: de fet, en l'assignatura ELEMENTS DE CAD se segueix utilitzant AutoCAD 2004, que correspon a les llicències adquirides en 2003 (a hores d'ara ja es comercialitza AutoCAD 2008), perquè fa tres anys i mig va optar per seguir amb aquest producte en comptes de passar-se amb tot l'equip a iCAD-T, que si no era pròpiament *software* lliure sí que era obert, en el sentit de donar accés al codi font. Conveniències personals a banda, que també n'hi havia (amortitzar l'esforç d'adaptació del material docent a les versions 13, 14 i 16, i poder dedicar-se a una producció més creativa, com la que prologa aquesta INTRODUCCIÓ), i una mica fart de la dinàmica en què havia entrat Autodesk (llançament cada any d'una "nova versió" amb més "novetats" en l'embolcall que en els continguts), si com s'havia anunciat la UPC ja no actualitzaria més aquestes llicències era qüestió d'esperar que l'alternativa iCAD-T evolucionés fins assolir unes prestacions equiparables a les d'AutoCAD 2004 (el nivell de la versió iCAD-T de sortida era comparable a la 14 d'AutoCAD), moment en què es procediria al relleu; mentrestant, els estudiants tenien dret a ser alliçonats en l'ús del producte més competitiu.

Aquesta declaració d'intencions dóna peu a l'autor a cloure la present INTRODUCCIÓ adreçant una invitació a la Beatriz Ruiz, professora de l'ETSAB i cap del Servei d'Aplicacions Informàtiques del COAC (entitat promotora a Espanya de iCAD-T), tot animant-la a entrar en el portal La Farga (https://lafarga.cpl.upc.edu/) i accedir a la resta del treball (aquestes primeres pàgines ja les haurà rebut per e-mail) i als quatre arxius .LSP que reprodueixen els blocs de codi enumerats a la pàgina 5 (COMPLEMENT, BLOC2D-SENSATR, BLOC-AMBATR i TRANSF-LINEAL, amb un cinquè arxiu de nom ENCAIXABLOCS, entre evocador i poca-solta, que en fa una descripció sumària), per jutjar si el software que s'hi ofereix pot ser de l'interès dels usuaris de la plataforma que els dos defensem, des de posicions i graus de compromís diferents, tot i constituir un avenç no específicament orientat cap al dibuix arquitectònic. Sabem que l'esforç d'adaptació seria mínim, i si més no per una vegada es podria evitar que, per la pròpia dinàmica del sistema, qualsevol aportació desinteressada contribuís a desequilibrar la balança a favor de Goliat i en detriment de David.

Part 1

CAMPS D'INSERCIONS EN 2D: 5 EXERCICIS COMENTATS



EXERCICI 1

Inserció de blocs, amb orientació constant (angle de rotació $\alpha = 0$), en què les dimensions **x** varien linealment amb I_x i les dimensions **y** varien linealment amb I_y (I_x , I_y són les coordenades cartesianes del punt d'inserció **I**).

Algebraicament, això s'expressa amb les equacions

$$x = A\ I_x + B$$
$$y = C\ I_y + D$$

Si no ens donen els coeficients **A** i **C** ni les constants **B** i **D**, n'hi haurà prou que ens subministrin les insercions realitzades a I_1 i I_2, sempre que sigui $I_{1x} \neq I_{2x}$ i $I_{1y} \neq I_{2y}$: amb les coordenades d'aquests punts i els valors x_1, x_2, y_1 i y_2, formaríem dos sistemes de dues equacions i dues incògnites cadascun, d'on trauríem **A**, **B**, **C** i **D**.

Doncs bé: l'exercici consisteix a definir el problema a partir de dues insercions i resoldre'l, però gràficament i no algebraica. Les tres qüestions a tractar són: donat **I**, determinació gràfica dels factors d'escala E_x i E_y (aspecte crític 1 de la Introducció); a partir d'aquests, determinació de les dimensions x_u i y_u del bloc que haurem d'utilitzar (aspecte 4) i creació d'aquest bloc; finalment, forma d'executar l'ordre ***INSERT*** (aspecte 6).

Determinació gràfica dels factors d'escala E_x i E_y

En general, no podem usar com a factors d'escala les coordenades I_x , I_y , perquè les longituds **x**,**y** no són proporcionals a aquests valors: si inserim el bloc just a l'origen ***SCP*** ($I_x = I_y = 0$), $x = B$ i $y = D$. Dit d'una altra manera:

- La coordenada O'_x, per a la qual $x = 0$, no és **0** sinó $-\frac{B}{A}$
- La coordenada O'_y, per a la qual $y = 0$, no és **0** sinó $-\frac{D}{C}$
- Així doncs, si en comptes de les coordenades ***SCP***, I_x i I_y, prenem les relatives al punt **O'**, $I'_x = I_x + \frac{B}{A}$ i $I'_y = I_y + \frac{D}{C}$, sí que podrem afirmar que les dimensions **x** i **y** són proporcionals a les coordenades homònimes del punt d'inserció, ja que les equacions queden reduïdes a

  $$x = A\left(I'_x - \frac{B}{A}\right) + B = A\ I'_x$$

  $$y = C\left(I'_y - \frac{D}{C}\right) + D = C\ I'_y$$

  En conseqüència, prendrem com a factors d'escala $E_x = I'_x$ i $E_y = I'_y$.

La determinació gràfica del punt **O'**, nou origen de coordenades, encara es més senzilla, com veiem a la Figura 1.1: el nou eix **X'** és l'horitzontal pel punt **OY'** on convergeixen les rectes que passen per dos parells de punts homòlegs (en cada element els punts han de tenir la mateixa coordenada **x**); el nou eix **Y'** és la vertical pel punt **OX'** on convergeixen les rectes que passen per dos parells de punts homòlegs (en cada element els punts han de tenir la mateixa coordenada **y**). En els dos elements subministrats hem escollit, com a punts homòlegs, els punts d'inserció I_1 i I_2, les puntes de fletxa **X** i les puntes de fletxa **Y**.

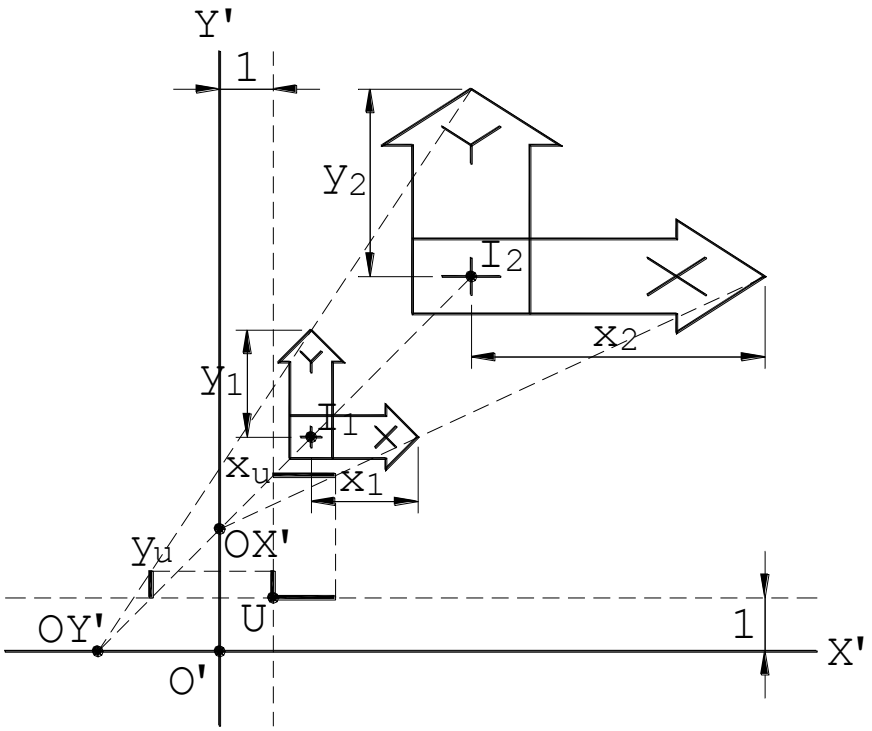


Figura 1.1

Determinació de les dimensions del bloc, x_u i y_u, i creació d'aquest bloc

Quan haguem d'inserir un bloc en el punt **I** usarem com a factors d'escala $E_x = I'_x$ i $E_y = I'_y$ (és a dir, les distàncies de **I** a l'eix **Y'** i a l'eix **X'**), ja que les dimensions **x** i **y** de la inserció varien proporcionalment a aquests valors, com acabem de veure. Ara bé: quines dimensions x_u i y_u haurà de tenir el bloc per tal que, en inserir-lo a I_1, per exemple, amb els factors d'escala $I'_{1x} = I_{1x} - O'_x$ i $I'_{1y} = I_{1y} - O'_y$, assoleixi precisament les dimensions x_1 i y_1? Doncs les que corresponen a les coordenades $I'_x = 1$, $I'_y = 1$ (és a dir, al punt de coordenades ***SCP*** $(O'_x + 1),(O'_y + 1)$): si anomenem x_u i y_u a les dimensions d'aquest bloc unitari, haurà de ser $\frac{x_1}{x_u} = \frac{I'_{1x}}{1}$ i $\frac{y_1}{y_u} = \frac{I'_{1y}}{1}$, és a dir, $x_1 = I'_{1x}\, x_u$ i $y_1 = I'_{1y}\, y_u$.

Gràficament, un cop dibuixats els eixos **X'** i **Y'** només caldrà treure'n còpies desplaçades una distància **1**, amb ***EQDIST***, i trobar les interseccions de la recta OY'– OX'–I_1–I_2 amb les línies obtingudes: el segment d'horitzontal comprès entre el punt d'intersecció amb la recta desplaçada de l'eix **Y'** i l'alineació dels punts $(I_{1x} + x_1), I_{1y}$ i $(I_{2x} + x_2), I_{2y}$ (a l'exemple, les dues puntes de fletxa **X**) representarà la dimensió x_u; el segment de vertical comprès entre el punt d'intersecció amb la recta desplaçada de l'eix **X'** i l'alineació dels punts $I_{1x}, (I_{1y} + y_1)$ i $I_{2x}, (I_{2y} + y_2)$ (les puntes de fletxa **Y**) representarà la dimensió y_u. Ara només cal, per facilitar la construcció del bloc, traslladar aquests segments sobre el eixos desplaçats, de manera que els extrems origen se situïn sobre el punt **U**.

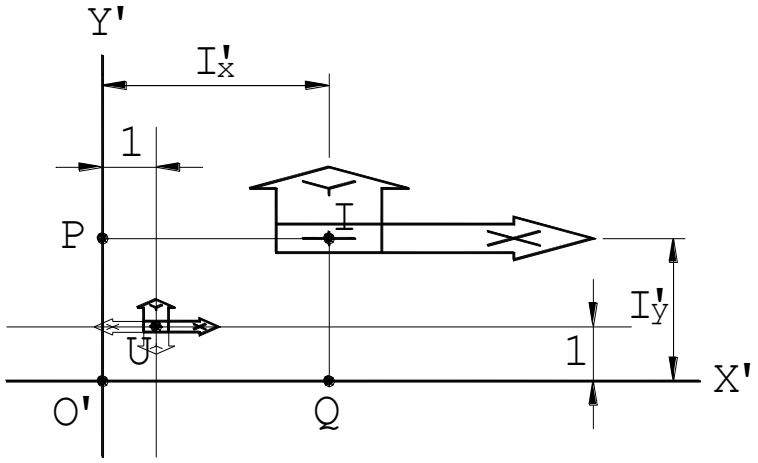
Atès que l'element que convertirem en bloc ha de tenir les dimensions x_u i y_u, el procediment dependrà de si ja disposem o no d'una figura regular, amb les dues dimensions iguals ($x_1 = y_1$, en el nostre exemple):

A) Si $x_1 = y_1$ o $x_2 = y_2$, copiem l'element regular en un lloc del dibuix lliure d'interferències i mitjançant l'ordre ***ESCALA*** (amb el punt d'inserció com a punt base i usant l'opció ***Referencia*** per determinar indirectament el factor d'escala: volem que la longitud x_1 o x_2 es converteixi en **1**) redimensionem la còpia de manera que faci 1×1.

B) Si $x_1 \neq y_1$ i $x_2 \neq y_2$, convertim en bloc un dels dos elements, el situat a I_2, per exemple (ATENCIÓ: l'element original no ha de desaparèixer). Inserirem aquest bloc en un lloc del dibuix lliure d'interferències i, usant com a factors d'escala $E_x = y_2$ i $E_y = x_2$, obtindrem una inserció regular (recordeu que per introduir gràficament les longituds y_2 i x_2 cal usar les opcions ***X*** i ***Y*** de ***INSERT***, abans d'entrar el punt d'inserció). Si explosionem aquesta inserció amb ***DESCOMP*** passarem a les condicions A i només caldrà usar ***ESCALA*** perquè la figura faci **1 × 1**.

Convertiu l'element **1 × 1** en bloc i inseriu-lo a **U**, prenent com a factors d'escala $E_x = x_u$ i $E_y = y_u$ (recordeu com cal utilitzar ***INSERT*** per fer això): ja teniu el bloc que necessitàveu. Només queda un petit detall: si executéssiu l'ordre ***BLOQUE*** seleccionant directament aquest element (que és una inserció de bloc) tindríeu un bloc allotjat dintre d'un altre. Un bloc així és perfectament operatiu, però simplificarem la base de dades AutoCAD (i, en salvar el dibuix, l'arxiu ***.DWG***) si abans l'explosionem amb ***DESCOMP***.

Figura 1.2



Inserció del bloc

Just unes línies enrera, tot descrivint el procés de creació del bloc, hem recordat dues vegades que per determinar gràficament els factors d'escala E_x i E_y calia "forçar" l'execució de ***INSERT***, postposant la introducció del punt d'inserció. Amb aquests comentaris encara calents, no tindria res d'estrany precipitar-se i pensar que, per inserir el bloc en una posició **I** qualsevol, no hi ha més remei que dibuixar a **I**, abans, un objecte punt (ja que necessitarem referir-nos-hi diverses vegades, sense poder recolzar-nos en la utilització de les coordenades **@**), activar amb caràcter permanent els modes de referència a objectes ***PUNto***, ***INTErsección*** i ***PERpendicular*** (cal deixar-ho tot preparat per fer front a nombroses insercions) i executar ***INSERT*** (usarem l'ordre ***-INSERT*** per facilitar-ne la transcripció) segons el guió següent:

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***X***
Precise factor de escala X: ***<clic sobre I>***
Designe segundo punto: ***<clic sobre P>***
Precise punto de inserción: ***Y***
Precise factor de escala Y: ***<clic sobre I>***
Designe segundo punto: ***<clic sobre Q>***
Precise punto de inserción: ***<clic sobre I>***
Precise ángulo de rotación <0>:

Noteu que hem necessitat de subministrar 9 ínputs per completar l'execució, i davant d'això convé preguntar-se si calia desglossar la introducció dels factors d'escala E_x = **I—P** i E_x = **I—Q**.

Realment no, i aquest exemple és l'únic dels que presentem aquí en què no cal alterar "l'ordre natural" d'introducció de dades, postposant el punt d'inserció: subministrar-les a mesura que ***INSERT*** ens les reclama és compatible amb la definició gràfica dels dos factors d'escala (d'un sol tret, a més!), fent innecessària l'ordre ***PUNTO*** i limitant a ***INTErsección*** l'activació amb caràcter permanent dels modes de referència a objectes. Hi ha dues possibilitats:

A) Si el bloc s'ha definit amb **U** com a punt base:

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]:
<clic sobre I>
Indique factor de escala X, precise esquina opuesta, o [Esquina/XYZ] <1>:
<clic sobre O′>
Precise ángulo de rotación <0>: ***180***

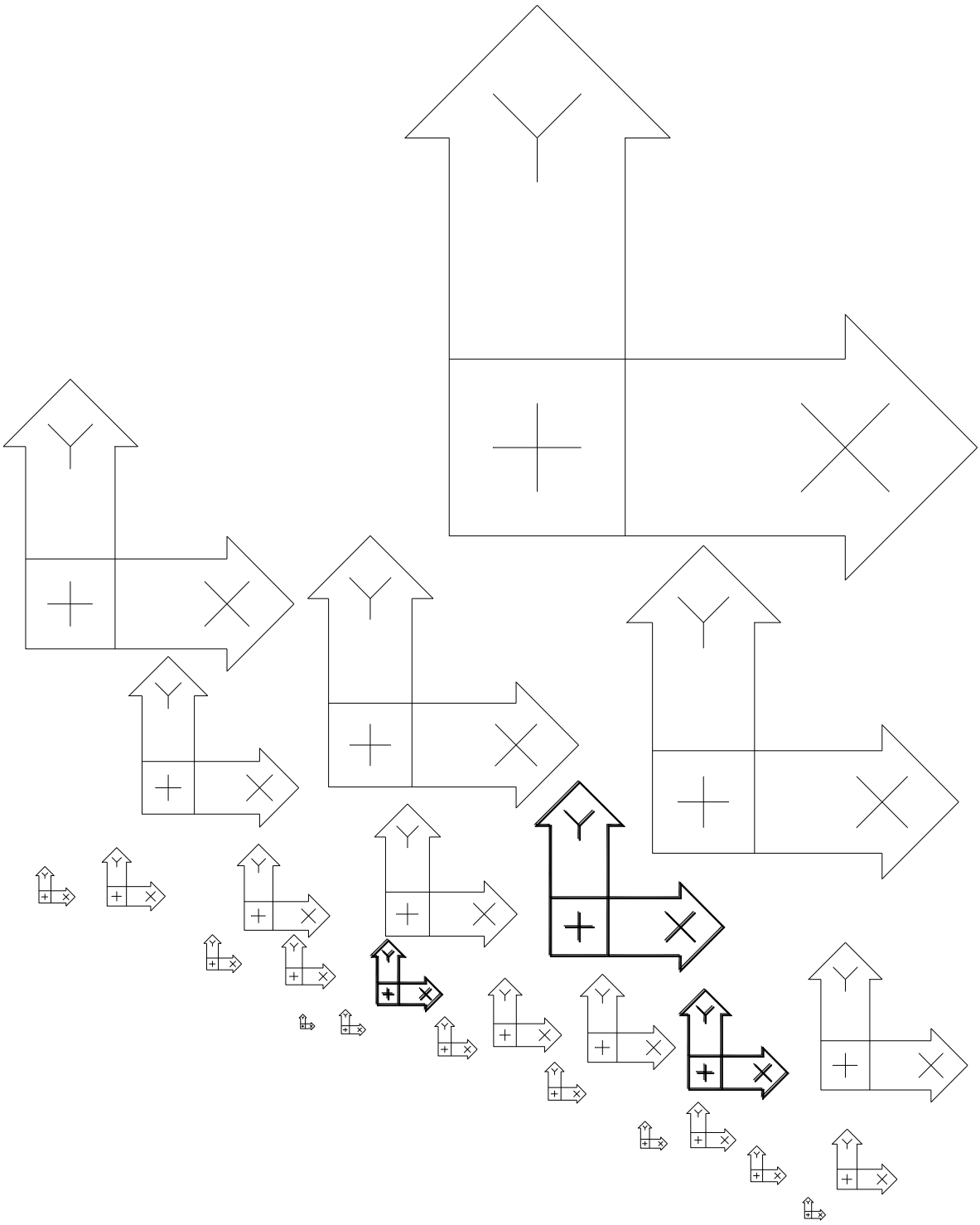
Si cal inserir amb un gir de **180°** és perquè els factors d'escala definits (components del vector **I→O′**) eren negatius. Ara bé, si el bloc l'haguéssim definit en posició invertida (deixant-lo girat **180°** quan vam inserir a **U** el bloc auxiliar **1 × 1**, com el dibuixat amb traç més fi a la Figura 1.2), ara podríem haver-lo inserit amb angle de **0°** (valor per defecte).

B) Si el bloc s'ha definit amb **O′** com a punt base:

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]:
<clic sobre O′>
Indique factor de escala X, precise esquina opuesta, o [Esquina/XYZ] <1>:
<clic sobre I>
Precise ángulo de rotación <0>:

En aquest cas hem pogut adoptar l'angle per defecte perquè els factors d'escala definits (components del vector **O′→I**) eren positius.

A banda de les simplificacions esmentades, és clar que amb A o B haurem baixat des de 9 ínputs fins a 4.



EXERCICI 2

Inserció de blocs, amb factor d'escala uniforme (en cada inserció, $E_x = E_y = E$) i orientació constant (angle de rotació $\alpha = 0$), en què la grandària l varia linealment amb I_x i amb I_y (I_x, I_y són les coordenades cartesianes del punt d'inserció I).

Algebraicament, això s'expressa amb l'equació

$$l = A\ I_x + B\ I_y + C$$

Si no ens donen els coeficients A i B ni la constant C, n'hi haurà prou que ens subministrin les insercions realitzades a I_1, I_2 i I_3, sempre que aquests tres punts no estiguin alineats (ha de ser $(I_{1x} - I_{2x})(I_{2y} - I_{3y}) \neq (I_{1y} - I_{2y})(I_{2x} - I_{3x})$): amb les coordenades d'aquests punts i els valors l_1, l_2 i l_3 (dimensions homòlogues en cadascuna de les tres insercions), formaríem un sistema de tres equacions amb les tres incògnites A, B i C.

Doncs bé: l'exercici consisteix a definir el problema a partir de tres insercions i resoldre'l, però gràficament i no algebraica. Les tres qüestions a tractar són: donat I, determinació gràfica del factor d'escala E (aspecte crític 1 de la Introducció); a partir d'aquest paràmetre, determinació de la grandària l_u del bloc que haurem d'utilitzar (aspecte 4) i creació d'aquest bloc; finalment, manera d'executar l'ordre ***INSERT*** (aspecte 6).

Determinació gràfica del factor d'escala E

De l'equació precedent es dedueix que:
- El lloc geomètric dels punts I en què $l = 0$ és la recta $A\ I_x + B\ I_y + C = 0$, que anomenarem recta de referència.
- El lloc geomètric dels punts I en què $l = l_n$ és la recta que passa per I_n ($l_n = A\ I_{nx} + B\ I_{ny} + C$), paral·lela a la de referència i separada d'ella una distància $d_n = \dfrac{l_n}{\sqrt{A^2 + B^2}}$, d'equació $A\ I_x + B\ I_y + (C - l_n) = 0$
- Així doncs, com que la grandària de les insercions,

  $$l = d\ \sqrt{A^2 + B^2}$$

  és proporcional a la distància d entre el punt d'inserció I i la recta de referència $A\ x + B\ y + C = 0$, adoptarem com a factor d'escala $E = d$.

En realitat, també podríem utilitzar com a factor d'escala distàncies no normals, és a dir, distàncies preses sobre rectes de direcció determinada, entre els punts d'inserció I i els d'intersecció amb la recta de referència $A\ x + B\ y + C = 0$. En aquest cas, si decidíssim de prendre distàncies d' en la direcció d'una recta d'equació $A'\ x + B'\ y = 0$ (evidentment, $\dfrac{A'}{B'} \neq \dfrac{A}{B}$), tindríem que

$$l = d'\ \frac{A\ B' - B\ A'}{\sqrt{A'^2 + B'^2}}$$

expressió de la qual la de més amunt respondria al cas particular en què la direcció escollida fos la perpendicular a la recta de referència ($\dfrac{A'}{B'} = -\dfrac{B}{A}$).
Únicament caldria precisar que, en haver variat el factor de proporcionalitat, la grandària l'_u d'un bloc per a insercions en què $E = d'$ hauria de ser diferent a la l_u d'un bloc per a insercions en què $E = d$ (exactament, $l'_u = l_u\ \dfrac{d}{d'}$).

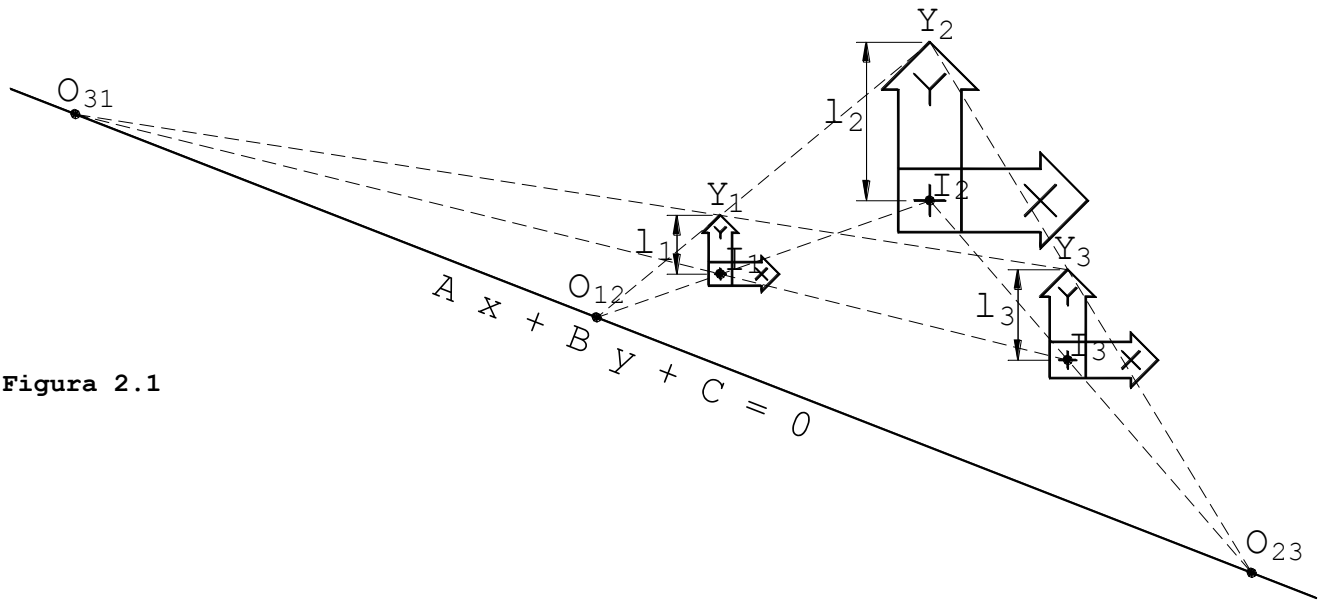


Figura 2.1

En plantejar el problema dintre d'AutoCAD, però, veurem que l'ús de direccions oblíqües a la recta $\mathbf{A\ x + B\ y + C = 0}$ no aporta cap avantatge, i que la distància normal és l'única que permet de generalitzar les insercions sense necessitat de construccions prèvies.

El primer pas, determinar gràficament la recta de referència $\mathbf{A\ x + B\ y + C = 0}$, no pot ser més senzill a partir de les tres insercions subministrades: el primer punt $\mathbf{O_{12}}$, és el de convergència de les rectes obtingudes unint dos parells de punts homòlegs de les insercions a $\mathbf{I_1}$ i $\mathbf{I_2}$ (per exemple, aquests mateixos punts i les respectives puntes de fletxa $\mathbf{Y_1}$ i $\mathbf{Y_2}$); el segon punt $\mathbf{O_{23}}$, és on ho fan les rectes obtingudes de les insercions a $\mathbf{I_2}$ i $\mathbf{I_3}$, pel mateix procediment; finalment, si es vol, un tercer punt $\mathbf{O_{31}}$, on convergeixen les rectes obtingudes aplicant el mateix procediment a $\mathbf{I_3}$ i $\mathbf{I_1}$, però només per comprovar que $\mathbf{O_{31}}$ està efectivament alineat amb $\mathbf{O_{12}}$ i $\mathbf{O_{23}}$. Si dues de les tres insercions tinguessin la mateixa grandària $\mathbf{l_i = l_j}$ (això implicaria que $\mathbf{\frac{I_{ix} - I_{jx}}{I_{iy} - I_{jy}} = -\frac{B}{A}}$) les dues rectes serien paral·leles (i paral·leles a $\mathbf{A\ x + B\ y + C = 0}$), i n'hi hauria prou a obtenir el punt de convergència determinat per les insercions de grandària $\mathbf{l_j \neq l_k}$ o $\mathbf{l_k \neq l_i}$ (ha de ser $\mathbf{\frac{I_{jx} - I_{kx}}{I_{jy} - I_{ky}} \neq -\frac{B}{A} \neq \frac{I_{kx} - I_{ix}}{I_{ky} - I_{iy}}}$) per dibuixar la recta de referència.

A partir d'aquí i tal com hem anunciat, en comptes d'adoptar com a factor d'escala la distància a la recta de referència (distància normal, volem dir), ja d'entrada, ens permetrem una petita digressió, amb l'objecte de veure com el plantejament analític de partida no era imprescindible i com, mitjançant aproximacions successives des d'una lògica geomètrica més directament lligada a l'ús d'AutoCAD, hauríem arribat al mateix lloc.

Suposem que, com a la Figura 2.1, el punt de convergència $\mathbf{O_{12}}$ és un punt propi ($\mathbf{l_1 \neq l_2}$, és a dir que les rectes que enllacen punts homòlegs de les insercions a $\mathbf{I_1}$ i $\mathbf{I_2}$ no són paral·leles ni ho són a la recta de referència), i posem-nos a buscar magnituds que tinguin a veure amb els punts d'inserció i amb les quals les respectives figures inserides mantinguin una relació mètrica constant, per tal d'usar-les com a factors d'escala:

1) Si, com a pas previ, restringíssim l'àmbit del problema, buscant un bloc i una manera d'inserir-lo aplicable només a punts **I** alineats amb $\mathbf{I_1}$ i $\mathbf{I_2}$, sens dubte que la primera idea que ens vindria al cap seria utilitzar una figura en què la longitud dels braços fos $\mathbf{l_u = 1}$ i adoptar com a factor d'escala la prevista per a l'element que hem d'inserir a **I**, $\mathbf{E = l}$. Tautològicament, $\mathbf{l = l_u\ E}$ (no fem

sinó afirmar que **l** és proporcional a **l**, amb factor de proporcionalitat **1**), però la dificultat és d'ordre pràctic: no només hem hagut de dibuixar la recta O_{12}—I_1—I_2 per situar-hi el punt **I**, sinó també la recta O_{12}—Y_1—Y_2, que convergeix amb la primera a O_{12}; però el més greu és que abans de cada inserció haurem d'elevar una línia vertical des de **I** fins a tallar aquesta segona, perquè no hi ha cap altra forma de poder avaluar la longitud **l**. (A la Figura 2.2 hem eliminat la inserció a I_1 i només s'ha deixat el punt O_{12} i la inserció a I_2, per tal de deixar prou espai a les successives propostes de bloc.)

2) Tot i mantenint-nos en la restricció enunciada, podríem eludir la necessitat de traçar una vertical prèvia a cada inserció. La clau d'aquest avenç consistiria a prendre la distància de **I** a la recta O_{12}—Y_2 perpendicularment i no vertical: ja no caldria dibuixar una línia abans de cada inserció, sinó que el mode ***PERpendicular*** de referència a objectes permetria l'avaluació directa de la distància normal $\overline{l}$. Tanmateix, nova identitat del factor d'escala implicaria la substitució del bloc usat abans per un altre de més gran: si adoptem $E = \overline{l}$, la longitud dels braços del bloc haurà de ser $l_u = \frac{l}{\overline{l}}$, i el procediment gràfic per obtenir una figura d'aquesta grandària constarà de dos passos:
   - Duplicar amb l'ordre ***COPIA*** una de les figures (p. ex., la inserida a I_2).
   - Amb l'ordre ***ESCALA***, escalar aquesta figura (podem fer-ho respecte a O_{12} o respecte a I_2). Com que desconeixem el factor d'escala necessari, haurem de recórrer a l'opció ***Referencia***, indicant que la longitud de referència actual, $\overline{l}_2$ (des de I_2, perpendicular a la recta O_{12}—Y_2), la volem convertir en **1**.

   La millora ha estat notable, però encara depenem de la recta O_{12}—Y_2: és cert que, de tota manera, havíem de dibuixar la recta Y_1—Y_2 per trobar el punt O_{12}, en el camí d'obtenció de la recta de referència $A\,x + B\,y + C = 0$, però si intentem extendre l'ús d'aquest bloc a punts no alineats amb I_1 i I_2 (de seguida, en el subapartat 4), l'obligació de carregar no només amb la recta O_{12}—I_2 sinó també amb la O_{12}—Y_2 ja esdevé inacceptable.

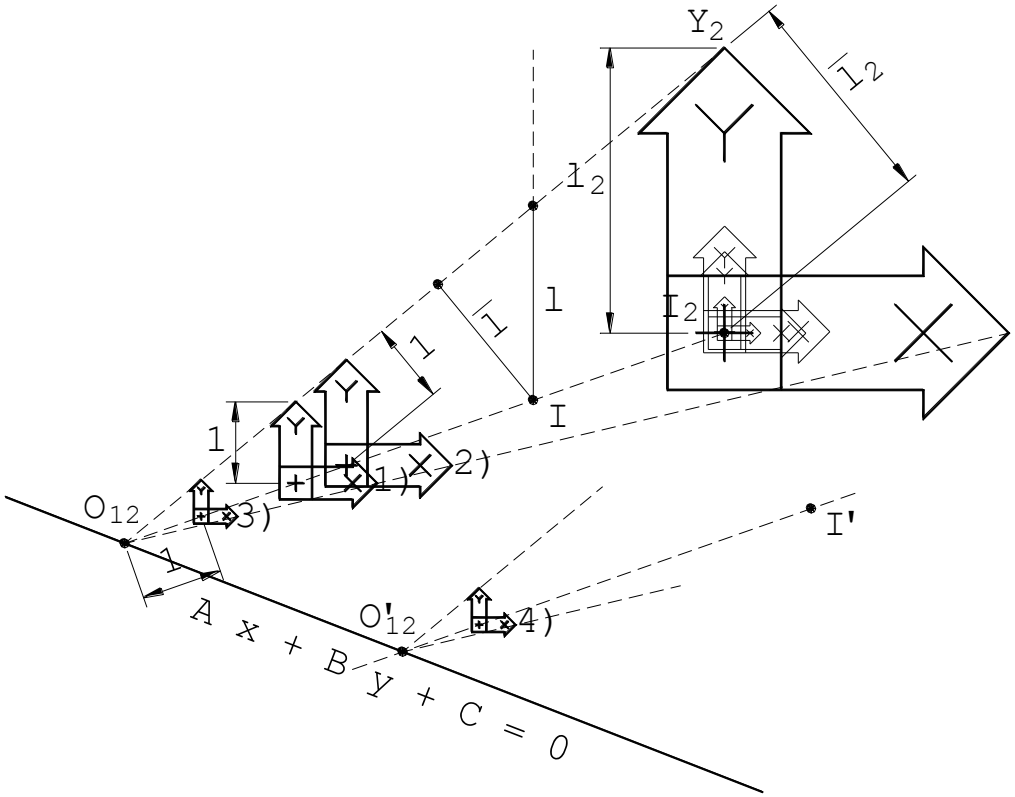
3) Mantenint encara la mateixa restricció (només insercions alineades amb O_{12} i I_2) però fent un esforç d'imaginació per treure'ns del damunt la recta O_{12}—Y_2, almenys pel que fa a la determinació del factor d'escala, ens adonem que hi ha una altra magnitud a la qual és proporcional la grandària **l** d'una inserció feta a **I**: la distància O_{12}—**I**. Així que, posats a usar com a factor d'escala aquesta distància, haurem de trobar el bloc adequat, de grandària $l_u = \frac{l_1}{O_{12}\text{—}I_1} = \frac{l_2}{O_{12}\text{—}I_2}$.

   Gràficament, seguirem el procediment habitual:
   - Duplicar amb ***COPIA*** una de les figures (per exemple, la inserida a I_2).
   - Amb ***ESCALA***, escalar-la (respecte a O_{12} o a I_2), recorrent a l'opció ***Referencia*** per manifestar que la longitud de referència actual, O_{12}—I_2, la volem veure convertida en **1**.

   Arribats aquí, podem considerar que hem optimitzat el sistema ... dintre de la severa restricció que suposa limitar-se a punts alineats amb O_{12} i I_2.

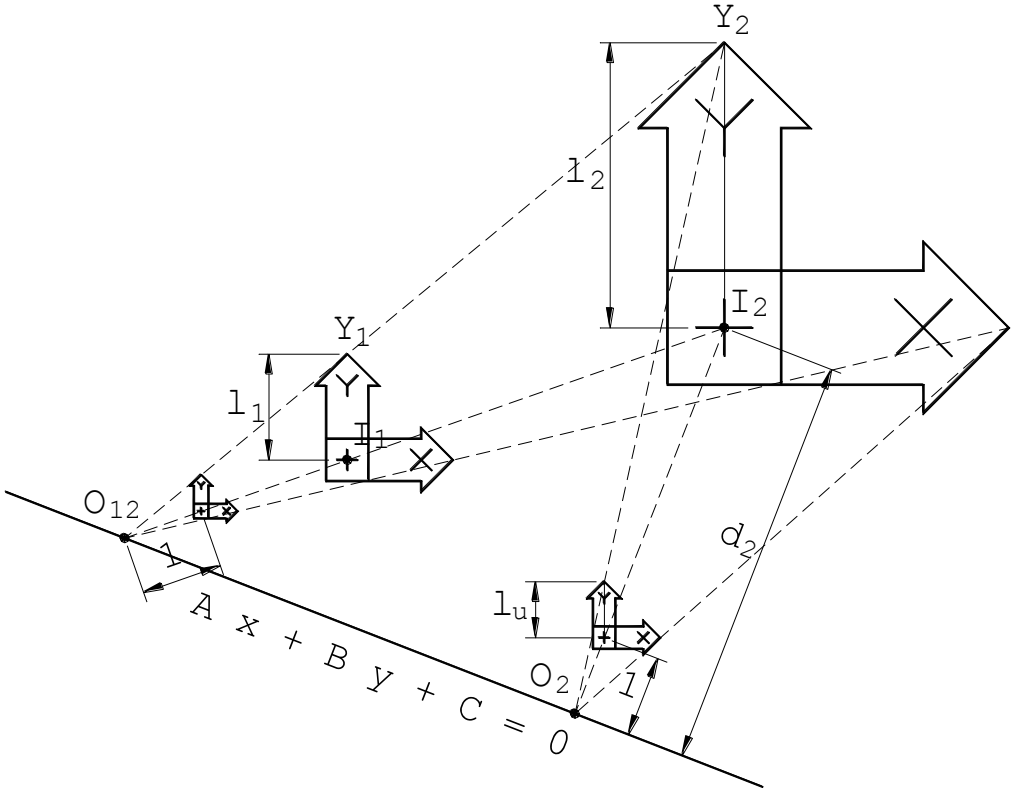
4) Ara és quan ens preguntem si el sistema té capacitat per transcendir aquest àmbit, extrapolant-lo a l'obtenció d'insercions no alineades amb I_1 i I_2. Doncs bé, sí que en té, i només hem d'imaginar que les manipulacions fetes a 1, 2 i 3 serien vàlides en qualsevol altra regió del pla, desplaçant paral·lelament a la recta de referència tota la construcció: les figures inicialment dibuixades a I_1 i I_2, amb les rectes O_{12}—Y_2 i O_{12}—I_2 (subapartats 1 i 2) o només amb l'última esmentada (subapartat 3). Però just en això, en haver de carregar sempre amb les línies de construcció, rau la poca versatilitat dels sistemes precedents: seguint el descrit a 3, abans d'inserir a I' hauríem de recórrer a ***EQDIST*** per traçar des d'aquest punt una paral·lela a O_{12}—I_2 i així poder usar com a factor d'escala la distància entre I' i la intersecció O'_{12} d'aquesta paral·lela amb la recta de referència (per assegurar una intersecció efectiva, l'original O_{12}—I_2 hauria de ser una recta il·limitada, dibuixada amb ***LINEAX***, i no pas un segment estricte dibuixat amb ***LINEA***); seguint el descrit a 2, a més, caldria recórrer per segon cop a ***EQDIST***, per traçar una paral·lela a O_{12}—Y_2 (també dibuixada amb ***LINEAX***) des d'aquesta intersecció. Com que després de cada inserció hauríem d'esborrar la paral·lela o parell de paral·leles (si la neteja l'anem deixant per al final, el dibuix es cobrirà progressivament amb una espessa trama), el balanç seria ben galdós: en comptes de resoldre cada inserció amb una sola ordre (***INSERT***), ho hauríem de fer amb tres o quatre (una o dues ***EQDIST***, ***INSERT*** i ***BORRA***).

Figura 2.2



Amb aquesta constatació clourem la digressió, perquè suposant que no haguéssim detectat d'entrada la conveniència d'usar com a factors d'escala la distància (normal) a la recta de referència **A x + B y + C = 0**, embrancant-nos en la línia discursiva 1-2-3-4, per força ens hauríem acabat fent la següent pregunta: si tot plantejament basat en blocs subordinats a direccions concretes **I_1–I_2**, **I_2–I_3**, **I_3–I_1** o altres derivades, implica el traçat previ de rectes en la mateixa direcció, per tal de poder avaluar el factor d'escala aplicable a cada inserció **I**, ¿hi ha alguna direcció privilegiada per a la qual la avaluació de distàncies pugui ser directa, prescindint d'aquest requisit? Naturalment, la resposta és que, un cop dibuixada la recta de referència **A x + B y + C = 0**, la direcció privilegiada és la seva normal, accessible mitjançant el mode ***PERpendicular*** de referència a objectes. Curiosament, en el subapartat 2 ja havíem descobert les excel·lències d'aquesta eina, però l'enfoc localista d'allà només ens havia permés de donar un petit pas.

Figura 2.3



Determinació de la dimensió del bloc, l_u, i creació d'aquest bloc

D'entrada, aclarirem que no cal dibuixar la recta I_2—O_2, perpendicular a la de referència (ni, per suposat, les que concorren a O_2 des de les puntes de fletxa **X** i **Y**): si s'ha fet a la Figura 2.3 només ha estat per il·lustrar millor l'aplicació del teorema de Tales, $\frac{l_u}{1} = \frac{l_2}{d_2}$, des d'on deduïm la grandària del bloc. Com ja havíem vist a l'apartat precedent, la figura escollida serà duplicada i escalada, i el procés culminarà en la seva conversió en bloc. Detallarem el guió d'aquestes operacions, però abans fem un suggeriment: si, malgrat ser innecessari dibuixar aquestes rectes (I_2—O_2 i les altres concorrents), no us importa fer-ho a canvi d'experimentar gràficament, visualitzant-ho "en animació", com la grandària **l** és proporcional a la distància **d**, només cal, en arribar a l'ordre ***ESCALA***, que executeu la subratllada com a opció alternativa en comptes de l'opció normal.

Comando: ***COPIA***
Designe objetos: ***<selecció del conjunt d'objectes que integren la figura>***
Designe objetos:
Precise punto base o de desplazamiento [Múltiple]: ***0,0***
Precise segundo punto del desplazamiento o <usar primero como desplazamiento>:

Comando: ***ESCALA*** <<<opció normal
Designe objetos: ***P***
Designe objetos:
Precise punto base: ***<clic sobre*** I_2 ***>***
Precise factor de escala o [Referencia]: ***R***
Precise longitud de referencia <1>: ***@***
Designe segundo punto: ***<clic sobre la recta de referència, amb el mode PERpendicular>***
Precise nueva longitud: ***1***

Comando: ***ESCALA*** <<<opció alternativa
Designe objetos: ***P***
Designe objetos:
Precise punto base: ***<clic sobre*** O_2 ***>***
Precise factor de escala o [Referencia]: ***R*** (però abans mou el cursor cap a I_2, poc a poc, per veure l'efecte virtual)
Precise longitud de referencia <1>: ***@***
Designe segundo punto: ***<clic sobre*** I_2 ***>***
Precise nueva longitud: ***1***

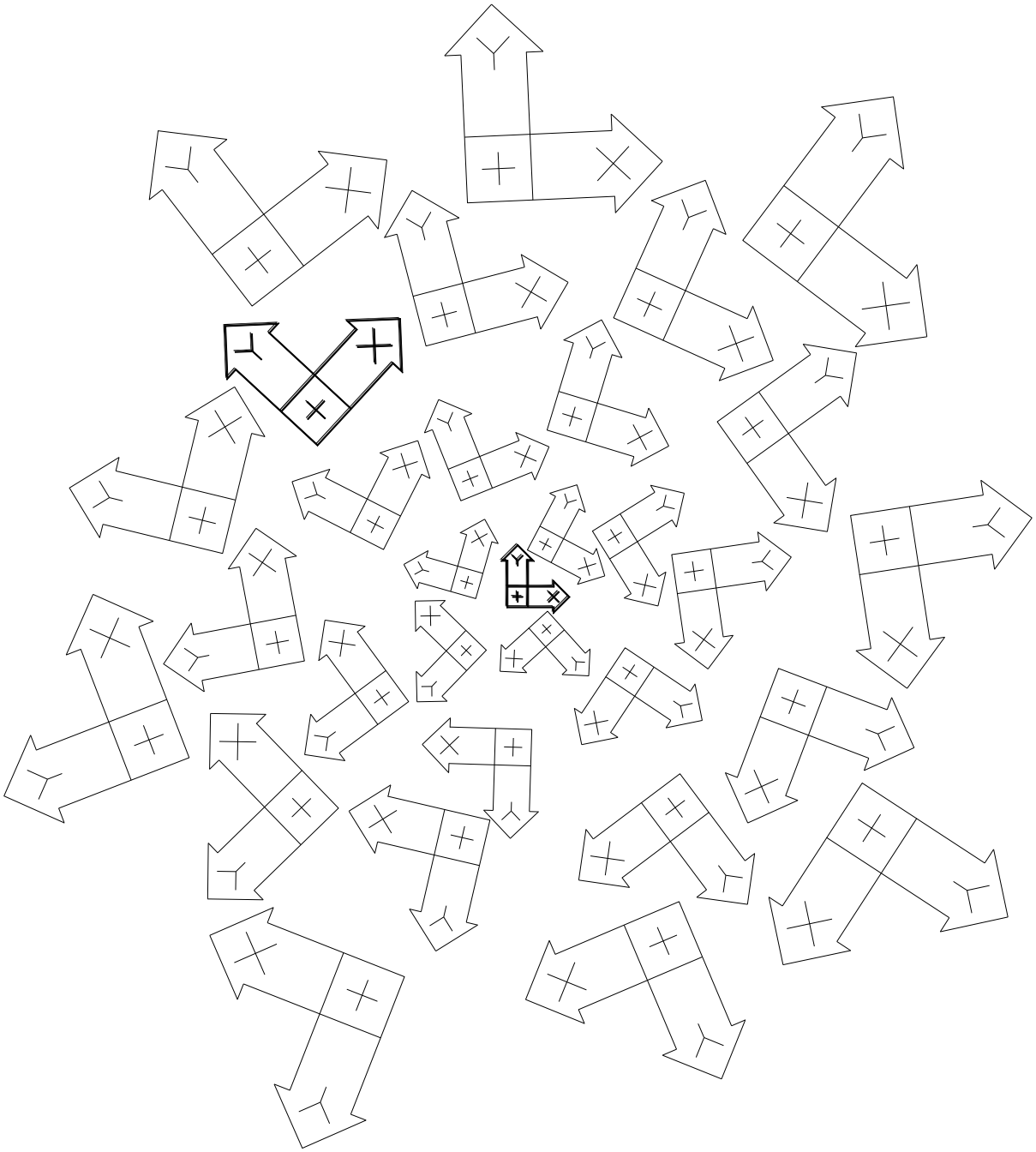
Comando: ***-BLOQUE***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto base de inserción: ***<clic sobre*** I_2 *>*
Designe objetos: ***P***
Designe objetos:

Inserció del bloc

Com havíem anunciat a la Introducció, cal realitzar les insercions sobre posicions consolidades d'alguna manera. Podem suposar, per exemple, que el mateix client que ens ha encomanat omplir el dibuix amb 500 figures que han de respondre a la llei deduïble de les tres ja subministrades, s'ha encarregat de dibuixar 497 objectes punt (ordre ***PUNTO***) en els llocs on les vol, a més de les 3 figures de referència (per cert que, si un cop dibuixada la recta de referència, resultés que alguns d'aquests punts se situen a l'altra banda, el client s'hauria de pronunciar sobre el particular, dient-nos si cal passar d'aquestes posicions, si cal inserir el bloc com a l'altra banda, sense afectació de l'angle de gir, o bé si cal fer-ho girant-lo 180°). Així doncs, en el guió que clou aquest exercici i que descriu la inserció del bloc a **I** (a diferència de l'Exercici 1, no hem cregut necessari

il·lustrar-la gràficament), suposarem que en aquesta posició ja hi tenim un objecte punt, però en consonància amb la notació usada fins ara només escriurem ***<clic sobre* I>**, assumint tàcitament que cal recórrer al mode ***PUNto*** de referència a objectes, i només farem explícita l'aplicació del mode ***PERpendicular*** a la recta de referència. A la pràctica, abans de procedir a les insercions recorreríem a ***REFENT*** per deixar activats amb caràcter permanent els modes ***PUNto*** i ***PERpendicular***.

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***E***
Precise factor de escala para los ejes XYZ: ***<clic sobre* I>**
Designe segundo punto: ***<clic sobre la recta de referència, amb el mode PERpendicular>***
Precise punto de inserción: ***<clic sobre* I>**
Precise ángulo de rotación <0>:



EXERCICI 3

Inserció de blocs, amb factor d'escala uniforme (en cada inserció, $\mathbf{E_x = E_y = E}$) i orientació constant en relació a un punt **O** (angle de rotació $\boldsymbol{\alpha = I_\varphi - 90^\circ - \alpha_0}$), on la grandària **l** varia linealment amb $\mathbf{I_r}$ ($\mathbf{I_r \geq 0}$ i $\mathbf{I_\varphi}$ són les coordenades polars del punt d'inserció **I** relatives al punt **O**).

Algebraicament, això s'expressa amb l'equació

$$\mathbf{l = A\ I_r + B}$$

on, a diferència de l'Exercici 4, suposem que **l** creix amb la distància $\mathbf{I_r}$ a **O** ($\mathbf{A > 0}$).

Si no ens donen el coeficient **A** ni la constant **B**, n'hi haurà prou que ens subministrin les insercions realitzades a $\mathbf{I_1}$ i $\mathbf{I_2}$, sempre que $\mathbf{I_{1r} \neq I_{2r}}$: amb les coordenades **r** d'aquests punts i els valors $\mathbf{l_1}$ i $\mathbf{l_2}$ (dimensions homòlogues en les dues insercions), formaríem un sistema de dues equacions amb les incògnites **A** i **B**.

Doncs bé: l'exercici consisteix a definir el problema a partir de dues insercions i resoldre'l, però gràficament i no algebraica. Les cinc qüestions a tractar són: donat **I**, determinació gràfica del factor d'escala **E** (aspecte crític 1 de la Introducció) i l'angle de rotació $\boldsymbol{\alpha}$ (aspecte 2); a partir d'aquests paràmetres, determinació de la grandària $\mathbf{l_u}$ (aspecte 4) i orientació $\boldsymbol{\alpha_0}$ (aspecte 5) del bloc que haurem d'utilitzar, i creació d'aquest bloc; finalment, manera d'executar l'ordre ***INSERT*** (aspecte 6).

Determinació gràfica del factor d'escala E

Primer de tot, mireu la Figura 3.1 i adoneu-vos que, en l'exercici actual, $\mathbf{I_1}$ coincideix amb el punt **O** ($\mathbf{I_{1r} = 0}$ i $\mathbf{I_{1\varphi}}$ resta indeterminat), és a dir, que un dels elements subministrats s'insereix precisament a l'origen de les coordenades polars que estructuren el conjunt, posició per a la qual no està definida l'orientació i que ens ha obligat a assignar-li arbitràriament el valor $\boldsymbol{\alpha = 0}$. Però també podria ser que cap dels dos elements subministrats s'inserís a **O** ($\mathbf{I_{1r} \neq 0 \neq I_{2r}}$): tot i així heu de seguir l'exposició en marxa, perquè al final de l'apartat ja donarem les pautes per poder-s'hi incorporar des d'altres supòsits.

Ni en general ni en el cas particular que tractem aquí, no podem usar com a factor d'escala la distància $\mathbf{I–O = I_r}$, perquè la grandària **l** no és proporcional a aquest valor: si inserim el bloc just a la posició **O** ($\mathbf{I_r = 0}$), la seva mida no és $\mathbf{l = 0}$ (tret que $\mathbf{B = 0}$) sinó que ha de coincidir amb la del primer element subministrat, $\mathbf{l_1 = A\ I_{1r} + B = B}$. Però si, en comptes de la distància $\mathbf{I_r}$ a **O**, considerem aquesta distància incrementada d'una longitud constant suplementària $\frac{\mathbf{B}}{\mathbf{A}}$, $\mathbf{d = I_r + \frac{B}{A}}$, sí que podrem parlar de proporcionalitat:

$$\mathbf{l = A\ I_r + B = A\left(I_r + \frac{B}{A}\right) = A\ d}$$

Si fos permès de considerar distàncies negatives (és a dir, punts d'inserció en què $\mathbf{I_r < 0}$), les insercions a una distància $\mathbf{I_r = -\frac{B}{A}}$ de **O** tindrien una grandària nul·la; en altres paraules, el lloc geomètric dels punts **I** en què $\mathbf{l = 0}$ seria una circumferència amb centre a **O** i de radi $\frac{\mathbf{B}}{\mathbf{A}}$. Gràficament, doncs, n'hi hauria prou a

dibuixar aquesta circumferència, que anomenarem cercle de referència, i per inserir la figura a qualsevol punt **I** prendríem com a factor d'escala $\mathbf{E = I_r + \frac{B}{A}}$, és a dir, la distància de **I** al punt més allunyat (la seva intersecció amb la prolongació de la línia radial **I–O**). Observeu que no caldria dibuixar la recta **I–O** (***LINEA***) ni molt menys allargar-la fins a tallar la circumferència (***ALARGA***), perquè primer faríem clic sobre **I** (amb el mode ***PUNto*** de referència a objectes activat, si les posicions d'inserció ens les subministren mitjançant objectes punt) i després sobre la part més allunyada de la circumferència, amb el mode ***PERpendicular*** activat (per un punt diferent del centre d'una circumferència sols hi passa una recta diametral).

Només hi ha una qüestió prèvia, i és com obtenir per procediments gràfics el valor $\mathbf{\frac{B}{A}}$ o, alternativament, un punt **O′** del cercle de referència (caracteritzats tots per donar lloc a insercions nul·les). Doncs ben senzill: considerant l'alineació $\mathbf{I_1–I_2}$, el seu punt **O′** serà on aquesta recta convergeixi amb la que resulta d'unir dos altres punts homòlegs de les figures subministrades.

Tanmateix, i tret que $\mathbf{I_{2\varphi} = 90^\circ}$, no hem d'oblidar que les figures subministrades no tenen la mateixa orientació i caldria que la tinguessin per tal de considerar-les homotètiques i poder parlar estrictament de punts homòlegs, de cara a la determinació del centre **O′** d'homotècia. Al respecte, podem fer tres coses:

- Duplicar la primera inserció (ATENCIÓ: l'element original no ha de desaparèixer) i girar-la un angle $\mathbf{\alpha_2 = I_{2\varphi} - 90^\circ}$ al voltant de $\mathbf{I_1}$ fins que quedi orientada com la segona.
- Duplicar la segona inserció (ATENCIÓ: l'element original no ha de desaparèixer) i girar-la un angle $\mathbf{90^\circ - I_{2\varphi}}$ al voltant de $\mathbf{I_2}$, de manera que $\mathbf{\alpha_2 - (90^\circ - I_{2\varphi}) = 0^\circ}$ i quedi orientada com la primera.
- Substituir ambdues figures per sengles cercles centrats a $\mathbf{I_1}$ i $\mathbf{I_2}$, de radi $\mathbf{l_1}$ i $\mathbf{l_2}$ respectivament. Aquests cercles els podríem dibuixar en una capa auxiliar i així, desactivant les demés, veuríem que n'és de senzill el problema del factor d'escala i de la grandària del bloc a utilitzar.

Aquest últim és el procediment representat a la Figura 3.1 i és el que recomanem, perquè es manté al marge de rotacions (l'aspecte d'un cercle no varia en girar-lo al voltant del seu centre) i per la fàcil identificació de punts homòlegs (punts quadrants d'igual orientació N, S, E o W, o punts de tangència $\mathbf{T_1}$ i $\mathbf{T_2}$ a una recta tangent comuna per l'exterior).

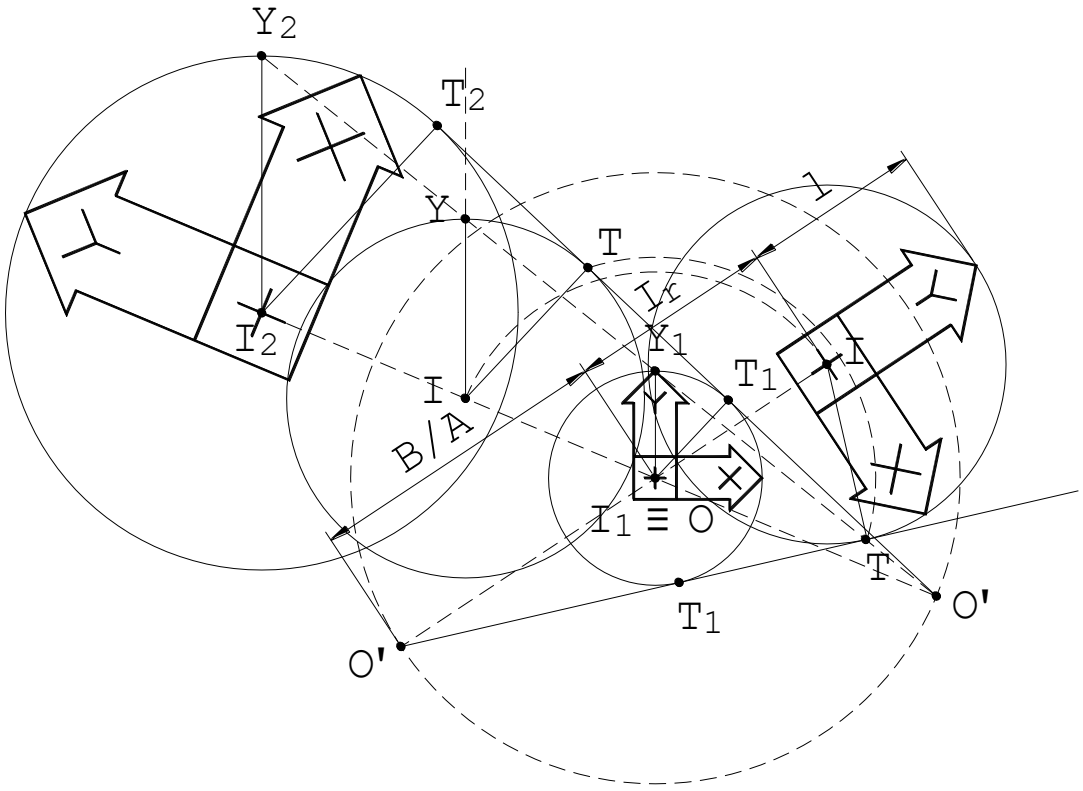


Figura 3.1

Queda per a més endavant la qüestió de quina mida l_u haurà de fer un bloc que s'insereixi prenent com a factor d'escala la distància augmentada $I_r + \frac{B}{A}$, però tal i com havíem fet a l'Exercici 2, abans de donar per tancat el tema ens permetrem un parèntesi, per valorar plantejaments alternatius en relació a la problemàtica del factor d'escala i la dimensió l_u.

Si ens centrem en les dues figures subministrades i de moment ens limitem a punts **I** alineats amb I_1 i I_2, ideant solucions que després miraríem de generalitzar a d'altres posicions, allò que sens dubte se'ns acudiria primer seria utilitzar com a bloc una figura en què els braços fessin $l_u = 1$ o, millor, considerar cercles circumscrits a les figures subministrades (per tal que l'orientació no interferís amb la qüestió de l'escala) i adoptar-ne un de radi **1** com a bloc, usant com a factor d'escala la longitud prevista per a la inserció. Tot i manipular cercles substituts, ben segur que en un primer moment prendríem el radi vertical o l'horitzontal com a element de referència (en definitiva, els braços de la figura substituïda), i això ens obligaria a traçar una línia vertical, per exemple, des del punt **I** fins a interceptar la recta O'–Y_1–Y_2, perquè no hi ha cap altra manera d'avaluar la longitud **l**. Però, per poc que seguíssim reflexionant, acabaríem veient que surt més a compte considerar els radis definits pels punts de tangència T_1 i T_2, l'avantatge indiscutible dels quals és que per avaluar distàncies ja no cal dibuixar cap recta abans de cada inserció: un cop tinguéssim O'–T_1–T_2, la longitud $l = I$–T la prendríem directament, amb el concurs del mode ***PERpendicular*** de referència a objectes.

Però aquest avenç es queda en no res quan provem d'extrapolar el mètode a punts **I** no alineats amb I_1 i I_2 (Figura 3.1, a la dreta i amunt de I_1): en cada **I** caldria copiar la tangent T_1–T_2 i girar-la un angle $I_\varphi - I_{2\varphi}$, per prendre-hi la distància $l = I$–T. Mirant de minimitzar les intervencions prèvies a cada inserció (l'últim que hem dit representa executar ***COPIA*** i ***GIRA*** abans de cada ordre ***INSERT***, sense comptar que després hauríem d'esborrar totes les còpies girades de T_1–T_2), també podríem haver pensat a prolongar les línies I_1–I_2 i T_1–T_2 fins a la intersecció O', i dibuixar el cercle de referència: així, abans d'inserir el bloc a cada nou punt **I** n'hi hagués hagut prou a executar ***LINEA*** per dibuixar una recta des de **I** fins al punt més llunyà d'aquest cercle (recta que passaria per $I_1 \equiv O$) i des d'aquí una tangent al cercle substitut centrat a I_1. Arribats aquí, gairebé segur que ens cauria la bena dels ulls i, disposant ja del cercle de referència, ens adonaríem que la solució més simple consisteix a usar com a factor d'escala la distància màxima I–O', proporcional a I–T per a qualsevol posició **I**.

Donem per acabat el parèntesi i seguim. Però abans d'ocupar-nos de la determinació gràfica de l'angle de rotació, encara hi havia un tema pendent: què passa quan $I_{1r} \neq 0 \neq I_{2r}$? Doncs bé, en aquest cas cal que ens explicitin el centre **O** o bé que ens localitzin a cada figura un punt Q_i alineat radialment amb el punt d'inserció I_i (no cal que Q_1 i Q_2 siguin homòlegs) per obtenir el centre com a intersecció de les rectes I_1–Q_1 i I_2–Q_2 (naturalment, això només serà possible si I_1 i I_2 no estan alineats radialment). A més, cal que s'acompleixin quatre condicions:

1) Ha de ser $l_1 \neq l_2$; altrament, el sistema restaria indeterminat (si $I_{1r} = I_{2r}$) o representaria el cas trivial en què el coeficient de l'equació és $A = 0$ i totes les insercions tenen la mateixa grandària $l = B$ (si $I_{1r} \neq I_{2r}$), supòsit exclòs dels Exercicis 3 i 4.
2) Com dèiem a l'enunciat, ha de ser $I_{1r} \neq I_{2r}$; altrament ($l_1 \neq l_2$ i $I_{1r} = I_{2r}$), el sistema seria incompatible.
3) Si $I_{ir} > I_{jr}$, ha de ser $l_i > l_j$; altrament, seríem en el domini de l'Exercici 4.
4) Les figures subministrades han de tenir la mateixa orientació relativa a **O**, requisit que es podria expressar dient que, si un punt R_1 de la primera està alineat amb O–I_1 (ordenació O–I_1–R_1, O–R_1–I_1 o R_1–O–I_1), el seu homòleg R_2 a la segona haurà d'estar alineat amb O–I_2 (mantenint l'ordenació O–I_2–R_2, O–R_2–I_2 o R_2–O–I_2, respectivament). Això només caldria verificar-ho si ens han explicitat el centre **O**; si l'hem deduït de dues alineacions radials I_1–Q_1 i I_2–Q_2, l'acompliment d'aquesta condició hi és implícita.

La determinació del cercle de referència dependrà de si les dues figures estan alineades radialment ($I_{1\varphi} = I_{2\varphi}$) o no (Figura 3.2): en el primer cas treballarem directament amb les figures subministrades, i en el segon n'haurem de substituir una per una còpia (ATENCIÓ: l'element original no ha de desaparèixer) girada al voltant de **O** fins a tenir I_1 i I_2 alineats amb aquest punt i a una mateixa banda.

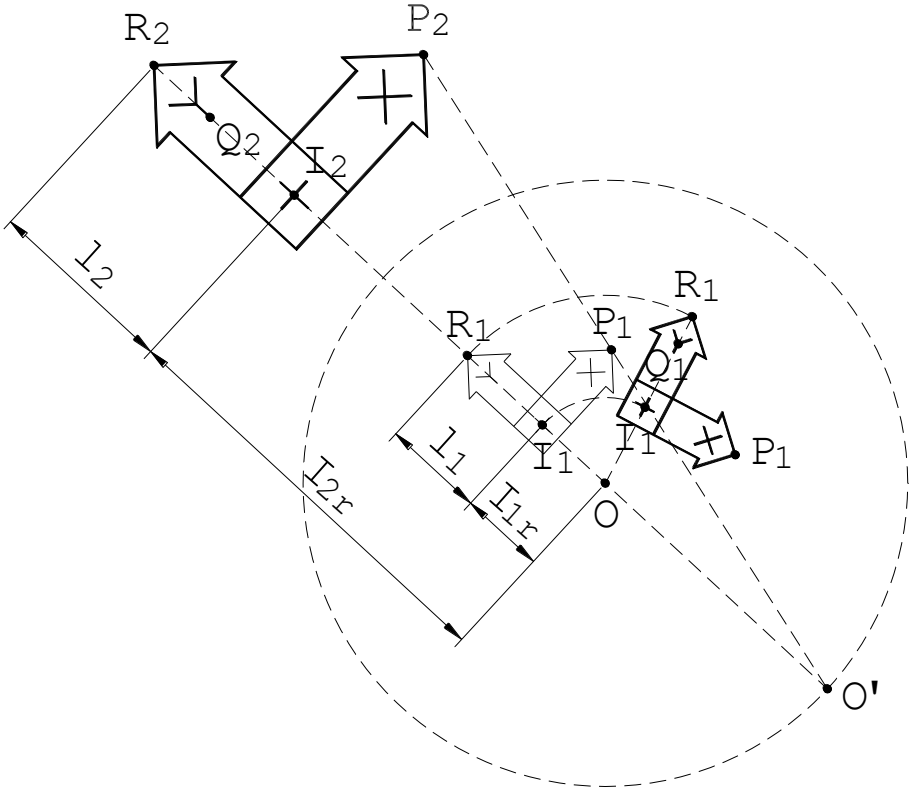


Figura 3.2

A partir d'aquí, localitzarem en les dues figures sengles punts **P**, homòlegs i tals que $P_\varphi \neq I_\varphi$, i n'obtindrem **O′** com a intersecció de les rectes $O\text{–}I_1\text{–}I_2$ i $P_1\text{–}P_2$. Com en el nostre exercici, dibuixarem el cercle de referència amb centre a **O** i passant per **O′**, però la manera de servir-nos-en pot diferir, segons on se situï **O′** en relació a I_1 i I_2, d'una banda, i a **O** de l'altra (recordem que $E = I_r + \frac{B}{A} = I\text{–}O'$):

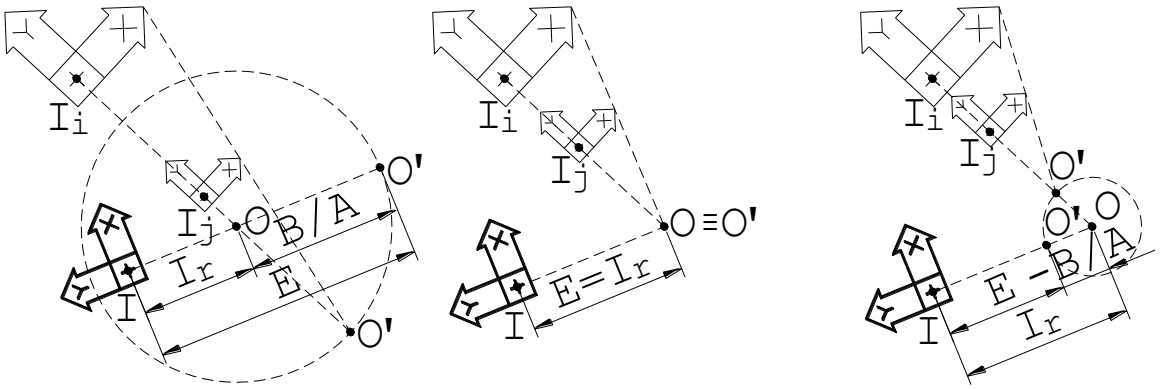


Figura 3.3

- Si l'ordenació dels punts és $I_i\text{–}I_j\text{–}O\text{–}O'$ (Figura 3.3, esquerra) ens mantenim en les condicions del nostre enunciat, les regles de joc estan definides per a qualsevol punt **I**, la constant **B** serà $B > 0$ i el factor d'escala $E = I\text{–}O' > I_r$.
- Si **O′** coincidís amb **O** (Figura 3.3, centre) tindríem una homotècia (llevat de la rotació), i seria $B = 0$ i $E = I_r$. A l'hora d'inserir, hauríem de referir-nos a **O** tant per a la determinació del factor d'escala com per a la de l'angle de gir.
- Si l'ordenació dels punts fos $I_i\text{–}I_j\text{–}O'\text{–}O$ (Figura 3.3, dreta) la constant **B** seria $B < 0$, les regles de joc només estarien definides per a posicions **I** en què $I_r \geq -\frac{B}{A}$ (el complement $-\frac{B}{A} \geq I_r > 0$, a l'altre banda del cercle de referència, ja pertany al domini de l'Exercici 4, on $A < 0$ i $B > 0$) i el factor d'escala seria $E = I\text{–}O' < I_r$. A l'hora d'inserir, en comptes de determinar el factor d'escala fent clic primer a **I** (amb el mode ***PUNto*** activat, si les posicions ens les subministren amb objectes punt dibuixats prèviament) i després sobre la regió més allunyada del cercle de referència, ho hauríem de fer sobre la regió més pròxima (però també amb el mode ***PERpendicular***).

Determinació gràfica de l'angle de rotació α

En una inserció a **I**, la figura (per ser més precisos, l'horitzontal de la figura en repòs, que en el nostre cas està representada pel braç o eix **X**) ha de quedar amb una inclinació $\mathbf{I_\varphi - 90^o}$. Si anomenem **α** l'angle de rotació utilitzat i α_0 la inclinació de la figura que hem adoptat com a bloc, la primera idea que se'ns acut és partir d'una figura en repòs, amb el braç **X** orientat cap a la dreta ($\alpha_0 = 0^o$), i inserir-la amb un angle de rotació $\alpha = \mathbf{I_\varphi - 90^o}$. Però de seguida, quan vulguem posar-la en pràctica, haurem de reconsiderar-la per poc operativa: així com l'orientació centrípeta o radial $\alpha = \mathbf{I_\varphi - 180^o}$ és immediata a partir del punt d'inserció **I** (amb el mode ***PUNto*** de referència a objectes, si **O** està representat per un objecte punt, o amb els modes ***CENtro*** o ***PERpendicular*** aplicats al cercle de referència), per definir l'orientació orbital o tangencial $\alpha = \mathbf{I_\varphi - 90^o}$, normal a l'anterior, hauríem de dibuixar abans una línia **I–P** perpendicular a **I–O** (de fet, la manera més senzilla d'obtenir-la és dibuixar **I–O** o **I–O′** i després girar-la **90°** al voltant de **I**), opció rebutjable sense cap mena de dubte, per la feinada que representa (passaríem de **n** execucions de ***INSERT*** a **n** ***INSERT***, **n** ***LINEA***, **n** ***GIRA*** i una execució de ***BORRA*** per eliminar les **n** línies auxiliars creades i girades). Així doncs, quan ***INSERT*** ens demani l'angle de rotació li subministrarem $\alpha = \mathbf{I_\varphi - 180^o}$ mitjançant alguns dels dispositius suggerits, i en l'apartat que ve tot seguit ja decidirem amb quina orientació cal definir el bloc partint d'aquesta premissa.

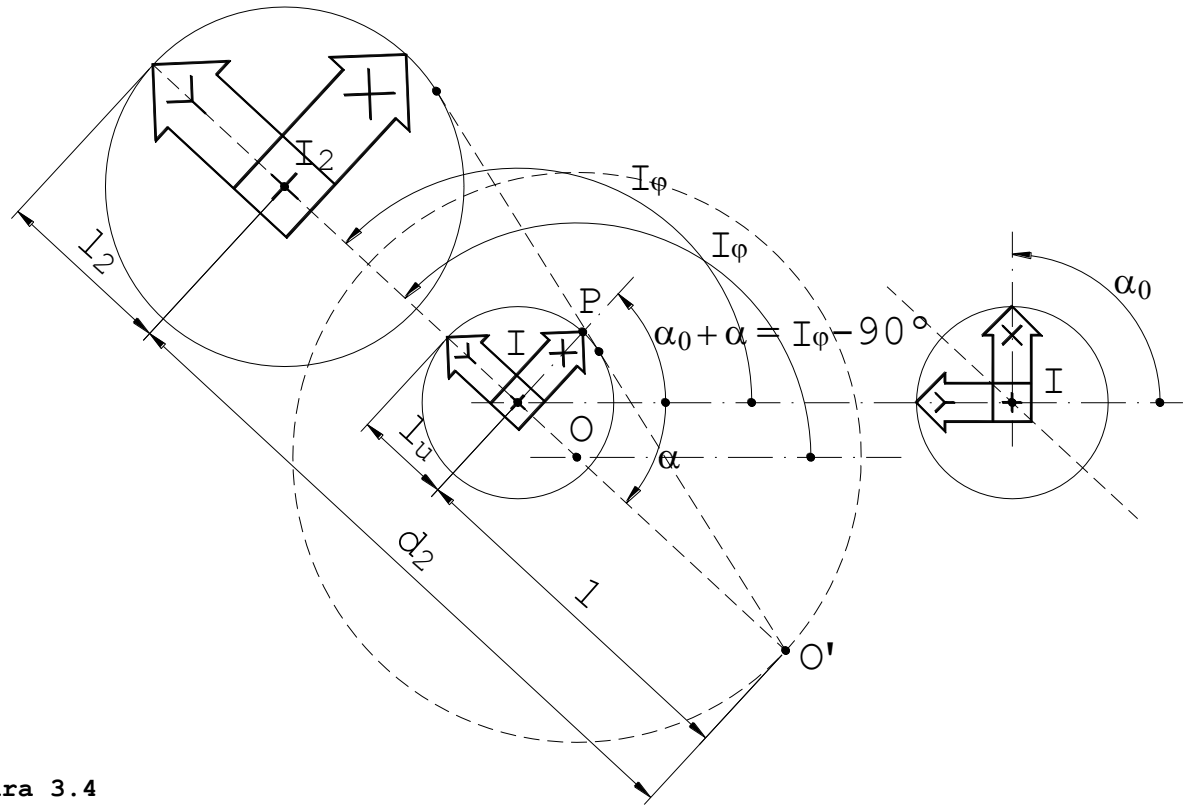


Figura 3.4

Determinació de la dimensió l_u i l'orientació α_0 del bloc. Creació d'aquest bloc

Si, dibuixat el cercle de referència, escollim una de les figures subministrades, per exemple la inserida a $\mathbf{I_2}$ (millor la més gran), el factor d'escala corresponent a aquesta inserció serà la distància $\mathbf{I_2–O'}$, que en la expressió

$$\mathbf{l_2 = A\ I_{2r} + B = A \left(I_{2r} + \frac{B}{A} \right) = A\ d_2}$$

està representada per $\mathbf{d_2 = I_{2r} + \frac{B}{A}}$. Això vol dir que la longitud $\mathbf{l_u}$ dels braços de la figura que adoptem com a bloc (o el radi del cercle substitut, si voleu) ha de ser $\mathbf{l_u = A = \frac{l_2}{d_2}}$. Com hem vist en l'Exercici 2, el procediment gràfic més directe per obtenir una figura $\mathbf{d_2}$ vegades més petita que la situada a $\mathbf{I_2}$ és definir-la com a homotètica d'aquesta respecte a $\mathbf{O'}$ i situar-la a una distància **1** d'aquest punt: en la pràctica, copiarem el model en la seva posició original i escalarem la còpia respecte a $\mathbf{O'}$, usant com a factor d'escala $\mathbf{\frac{1}{l_2}}$:

Comando: ***COPIA***
Designe objetos: ***<selecció del conjunt d'objectes que integren la figura>***
Designe objetos:
Precise punto base o de desplazamiento [Múltiple]: ***0,0***
Precise segundo punto del desplazamiento o <usar primero como desplazamiento>:

Comando: ***ESCALA***
Designe objetos: ***P***
Designe objetos:
Precise punto base: ***<clic sobre O'>***
Precise factor de escala o [Referencia]: ***R***
Precise longitud de referencia <1>: ***@***
Designe segundo punto: ***<clic sobre I_2>***
Precise nueva longitud: ***1***

Quant a l'orientació $\mathbf{\alpha_0}$ de la figura que utilitzarem com a bloc, partirem del que havíem convingut a l'apartat precedent (inserir-lo amb un angle $\mathbf{\alpha = I_\varphi - 180^\circ}$, que equival a apuntar cap al centre **O**, simplement perquè això era més fàcil de fer) i, tenint en compte que ha d'acomplir-se que $\mathbf{\alpha_0 + \alpha = I_\varphi - 90^\circ}$ (o $\mathbf{\alpha = I_\varphi - 90^\circ - \alpha_0}$, com escrivíem en el quadre inicial), ara ens tocarà decidir entre dues possibilitats:
- Si fem $\mathbf{\alpha_0 = 0^\circ}$, la figura quedarà inserida amb una inclinació $\mathbf{\alpha_0 + \alpha = I_\varphi - 180^\circ}$ (el braç **X** assenyalarà cap al centre **O**) i caldrà donar-li un gir addicional de $\mathbf{90^\circ}$ al voltant de **I** per tal d'ajustar-lo a $\mathbf{I_\varphi - 90^\circ}$.
- Si fem $\mathbf{\alpha_0 = 90^\circ}$, la figura quedarà inserida amb una inclinació $\mathbf{\alpha_0 + \alpha = I_\varphi - 90^\circ}$ (el braç **Y** adoptarà una orientació centrífuga en relació al centre **O**) i ja no caldrà tocar-la.

Obviament, la segona opció és la correcta, perquè la primera ens obligaria a executar $\mathbf{2\,n}$ ordres (**n *INSERT*** i **n *GIRA***) per deixar les **n** insercions correctament orientades, tot i que això representi un avenç respecte a les $\mathbf{3\,n + 1}$ ordres que hagués comportat inserir segons direccions orbitals (apartat precedent). Així doncs, un cop girada la figura unitària ja podrem fer-ne un bloc:

Comando: ***GIRA***
Angulo actual positivo en SCP: ANGDIR=sentido horario inverso ANGBASE=0
Designe objetos: ***P***
Designe objetos:
Precise punto base: ***<clic sobre I>***
Precise ángulo de rotación o [Referencia]: ***R***
Precise ángulo de referencia <0>: ***@***
Designe segundo punto: ***<clic sobre P>***
Precise el nuevo ángulo: ***90***

Comando: ***-BLOQUE***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto base de inserción: ***<clic sobre I>***
Designe objetos: ***P***
Designe objetos:

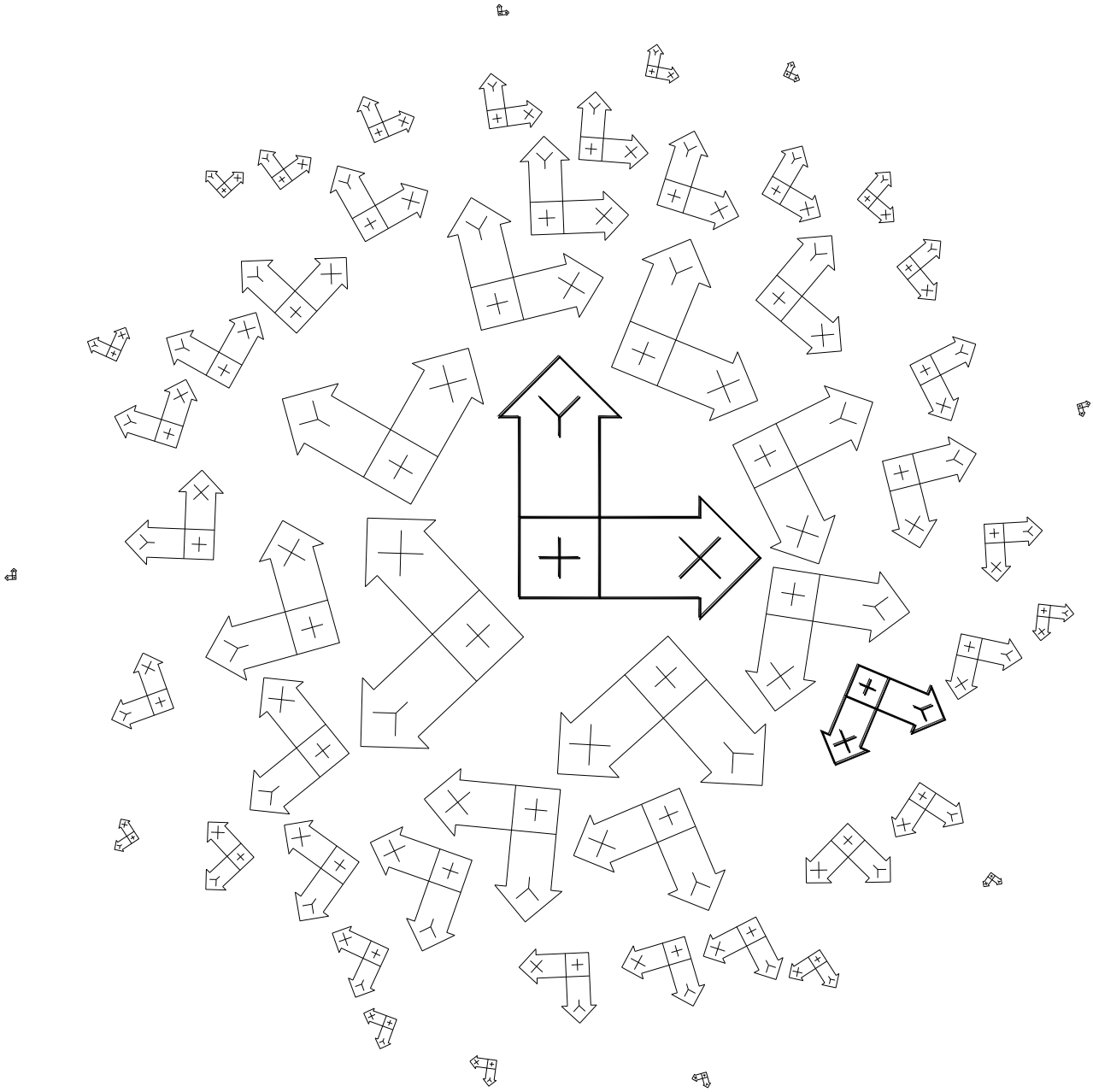
Abans de passar al següent apartat per veure la mecànica de la inserció d'un bloc de què ja coneixem grandària i orientació, seria bo tornar sobre aquest darrer tema per copsar el significat últim del gir $\mathbf{\alpha_0}$ i poder generalitzar l'estratègia de resolució a qualsevol altre cas: quan, per alguna raó (perquè és més còmode o, simplement, perquè no hi ha alternativa), decidim que a l'hora d'inserir un bloc cal definir gràficament l'angle de rotació $\mathbf{\alpha}$ en una direcció determinada, que pot coincidir o no amb la direcció en què volem que quedi orientat el bloc (per ser més precisos, l'eix **X** associat al bloc), aquest haurà d'estar girat un angle $\mathbf{\alpha_0}$

tal que els dos punts utilitzats en la determinació gràfica de **α** (el primer dels quals acostuma a coincidir amb **I**) quedin alineats horitzontalment, amb el segon a la dreta del primer (**I—P**, a la Figura 3.4). En aquest sentit, pot ser útil donar-li la volta a un enunciat gairebé tautològic, per disposar d'una regla pràctica sobre les característiques que ha de reunir la figura que anem a utilitzar com a bloc: com que quan inserim un bloc amb factors d'escala $\mathbf{E_x = E_y = 1}$ i amb angle de rotació $\mathbf{\alpha = 0^o}$ obtenim una reproducció exacta de la figura que havíem convertit en bloc (només caldria explosionar-la amb ***DESCOMP*** perquè la rèplica fos igual a l'original, no només pel que fa a l'aparença visual sinó també estructuralment), a l'hora d'esbrinar com haurà de ser un bloc per tal que, en inserir-lo com havíem convingut (determinant gràficament els factors d'escala i l'angle de gir amb unes longituds i una direcció fàcils de situar sobre el dibuix) quedi com volem, només hem d'imaginar-lo inserit utilitzant longituds de **1 Ud.** i una direcció de **0°**.

Inserció del bloc

Com havíem anunciat a la Introducció, cal realitzar les insercions sobre posicions consolidades d'alguna manera. Podem suposar, per exemple, que el mateix client que ens ha encomanat omplir el dibuix amb 500 figures que han de respondre a la llei deduïble de les dues ja subministrades, s'ha encarregat de dibuixar 498 objectes punt (ordre ***PUNTO***) en els llocs on les vol, a més de les 2 figures de referència. Així doncs, en el guió que clou aquest exercici i que descriu la inserció del bloc a **I** suposarem que en aquesta posició hi tenim un objecte punt, però en consonància amb la notació usada fins ara només escriurem ***<clic sobre* I>**, assumint tàcitament que cal recórrer al mode ***PUNto*** de referència a objectes, i només farem explícita l'aplicació del mode ***PERpendicular*** al cercle de referència. A la pràctica, abans de procedir a les insercions recorreríem a ***REFENT*** per deixar activats amb caràcter permanent els modes ***PUNto*** i ***PERpendicular***. No n'activarem cap més, perquè aquest últim ens pot servir tant per al factor d'escala com per a l'angle de rotació, anant a buscar en ambdós casos la posició del cercle de referència més allunyada de **I**: encara que el centre **O** no estigui representat per cap objecte punt (cas en què hi podríem fer clic directament), activar també el mode ***CENtro*** només per orientar el bloc seria una complicació, ja que quan no poséssim massa cura en la determinació del factor d'escala, mirant de no separar-nos massa del punt del cercle de referència diametralment oposat a **I**, l'acció de ***CENtro*** podria adquirir prioritat sobre la de ***PERpendicular*** i la posició referenciada seria **O** en comptes de **O'**.

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***E***
Precise factor de escala para los ejes XYZ: ***<clic sobre* I>**
Designe segundo punto: ***<clic sobre el punt més allunyat del cercle de referència, amb el mode PERpendicular>***
Precise punto de inserción: ***<clic sobre* I>**
Precise ángulo de rotación <0>: ***<clic sobre el punt més allunyat del cercle de referència, amb el mode PERpendicular>***



EXERCICI 4

Inserció de blocs, amb factor d'escala uniforme (en cada inserció, $\mathbf{E_x = E_y = E}$) i orientació constant en relació a un punt **O** (angle de rotació $\boldsymbol{\alpha = I_\varphi - 90^\circ - \alpha_0}$), on la grandària **l** varia linealment amb $\mathbf{I_r}$ ($\mathbf{I_r \geq 0}$ i $\mathbf{I_\varphi}$ són les coordenades polars del punt d'inserció **I** relatives al punt **O**).

Algebraicament, això s'expressa amb l'equació

$$\mathbf{l = A\ I_r + B}$$

on, a diferència de l'Exercici 3, suposem que **l** decreix amb la distància $\mathbf{I_r}$ a **O** ($\mathbf{A < 0}$ i $\mathbf{B > 0}$).

Si no ens donen el coeficient **A** ni la constant **B**, n'hi haurà prou que ens subministrin les insercions realitzades a $\mathbf{I_1}$ i $\mathbf{I_2}$, sempre que $\mathbf{I_{1r} \neq I_{2r}}$: amb les coordenades **r** d'aquests punts i els valors $\mathbf{l_1}$ i $\mathbf{l_2}$ (dimensions homòlogues en les dues insercions), formaríem un sistema de dues equacions amb les incògnites **A** i **B**.

Doncs bé: l'exercici consisteix a definir el problema a partir de dues insercions i resoldre'l, però gràficament i no algebraica. Les cinc qüestions a tractar són: donat **I**, determinació gràfica del factor d'escala **E** (aspecte crític 1 de la Introducció) i l'angle de rotació $\boldsymbol{\alpha}$ (aspecte 2); a partir d'aquests paràmetres, determinació de la grandària $\mathbf{l_u}$ (aspecte 4) i orientació $\boldsymbol{\alpha_0}$ (aspecte 5) del bloc que haurem d'utilitzar, i creació d'aquest bloc; finalment, manera d'executar l'ordre ***INSERT*** (aspecte 6).

Determinació gràfica del factor d'escala E

Primer de tot, mireu la Figura 4.1 i adoneu-vos que, en l'exercici actual, $\mathbf{I_1}$ coincideix amb el punt **O** ($\mathbf{I_{1r} = 0}$ i $\mathbf{I_{1\varphi}}$ resta indeterminat), és a dir, que un dels elements subministrats s'insereix precisament a l'origen de les coordenades polars que estructuren el conjunt, posició per a la qual no està definida l'orientació i que ens ha obligat a assignar-li arbitràriament el valor $\boldsymbol{\alpha = 0}$. Però també podria ser que cap dels dos elements subministrats s'inserís a **O** ($\mathbf{I_{1r} \neq 0 \neq I_{2r}}$): tot i així heu de seguir l'exposició en marxa, perquè al final de l'apartat ja donarem les pautes per poder-s'hi incorporar des d'altres supòsits.

Ni en general ni en el cas particular que tractem aquí, no podem usar com a factor d'escala la distància $\mathbf{I\text{–}O = I_r}$, perquè la grandària **l** no és proporcional a aquest valor: si inserim el bloc just a la posició **O** ($\mathbf{I_r = 0}$), la seva mida no és $\mathbf{l = 0}$ (tret que $\mathbf{B = 0}$) sinó que ha de coincidir amb la del primer element subministrat, $\mathbf{l_1 = A\ I_{1r} + B = B}$. Però si, en comptes de la distància $\mathbf{I_r}$ a **O**, considerem el complement o diferència de $\mathbf{I_r}$ a una longitud constant $\boldsymbol{-\frac{B}{A}}$ (com que $\mathbf{B > 0}$, ha de ser $\boldsymbol{-\frac{B}{A} \geq I_r > 0}$), $\boldsymbol{d = -\frac{B}{A} - I_r = -\left(I_r + \frac{B}{A}\right) \geq 0}$, sí que hi haurà proporcionalitat:

$$\mathbf{l = A\ I_r + B = A\left(I_r + \frac{B}{A}\right) = -\ A\ d}$$

D'acord amb això, les insercions a una distància $\boldsymbol{I_r = -\frac{B}{A}}$ de **O** tindran una grandària nul·la; en altres paraules, el lloc geomètric dels punts **I** en què $\mathbf{l = 0}$ és una circumferència amb centre a **O** i radi $\boldsymbol{-\frac{B}{A}}$. Gràficament, doncs, n'hi hauria

prou a dibuixar aquesta circumferència, que anomenarem cercle de referència, i per inserir la figura a qualsevol punt **I** prendríem com a factor d'escala una longitud $\mathbf{E = -\left(I_r + \frac{B}{A}\right)}$, és a dir, la distància de **I** al punt més proper (la intersecció amb la línia radial **O–I**). No cal dibuixar la recta **O–I** (***LINEA***) ni molt menys allargar-la fins a tallar la circumferència (***ALARGA***), perquè primer podem fer clic sobre **I** (amb el mode ***PUNto*** de referència a objectes activat, si les posicions d'inserció ens les donen mitjançant objectes punt) i després sobre la part més propera de la circumferència, amb el mode ***PERpendicular*** activat (per un punt diferent del centre d'una circumferència sols hi passa una recta radial).

Només hi ha una qüestió prèvia, i és com obtenir per procediments gràfics el valor $\mathbf{-\frac{B}{A}}$ o, alternativament, un punt **O′** del cercle de referència (caracteritzats tots per donar lloc a insercions nul·les). Doncs ben senzill: considerant l'alineació $\mathbf{I_1–I_2}$, el seu punt **O′** serà on aquesta recta convergeixi amb la que resulta d'unir dos altres punts homòlegs de les figures subministrades.

Tanmateix, i tret que $\mathbf{I_{2\varphi} = 90°}$, no hem d'oblidar que les figures subministrades no tenen la mateixa orientació i caldria que la tinguéssin per tal de considerar-les homotètiques i poder parlar estrictament de punts homòlegs, de cara a la determinació del centre **O′** d'homotècia. Al respecte, podem fer tres coses:

- Duplicar la primera inserció (ATENCIÓ: l'element original no ha de desaparèixer) i girar-la un angle $\mathbf{\alpha_2 = I_{2\varphi} - 90°}$ al voltant de $\mathbf{I_1}$ fins que quedi orientada com la segona.
- Duplicar la segona inserció (ATENCIÓ: l'element original no ha de desaparèixer) i girar-la un angle $\mathbf{90° - I_{2\varphi}}$ al voltant de $\mathbf{I_2}$, de manera que $\mathbf{\alpha_2 - (90° - I_{2\varphi}) = 0°}$ i quedi orientada com la primera.
- Substituir ambdues figures per sengles cercles centrats a $\mathbf{I_1}$ i $\mathbf{I_2}$, de radi $\mathbf{l_1}$ i $\mathbf{l_2}$ respectivament. Aquests cercles els podríem dibuixar en una capa auxiliar i així, desactivant les demés, veuríem que n'és de senzill el problema del factor d'escala i de la grandària del bloc a utilitzar.

Aquest últim és el procediment representat a la Figura 4.1 i és el que recomanem, perquè es manté al marge de rotacions (l'aspecte d'un cercle no varia en girar-lo al voltant del seu centre) i per la fàcil identificació de punts homòlegs (punts quadrants d'igual orientació N, S, E o W, o punts de tangència $\mathbf{T_1}$ i $\mathbf{T_2}$ a una recta tangent comuna per l'exterior).

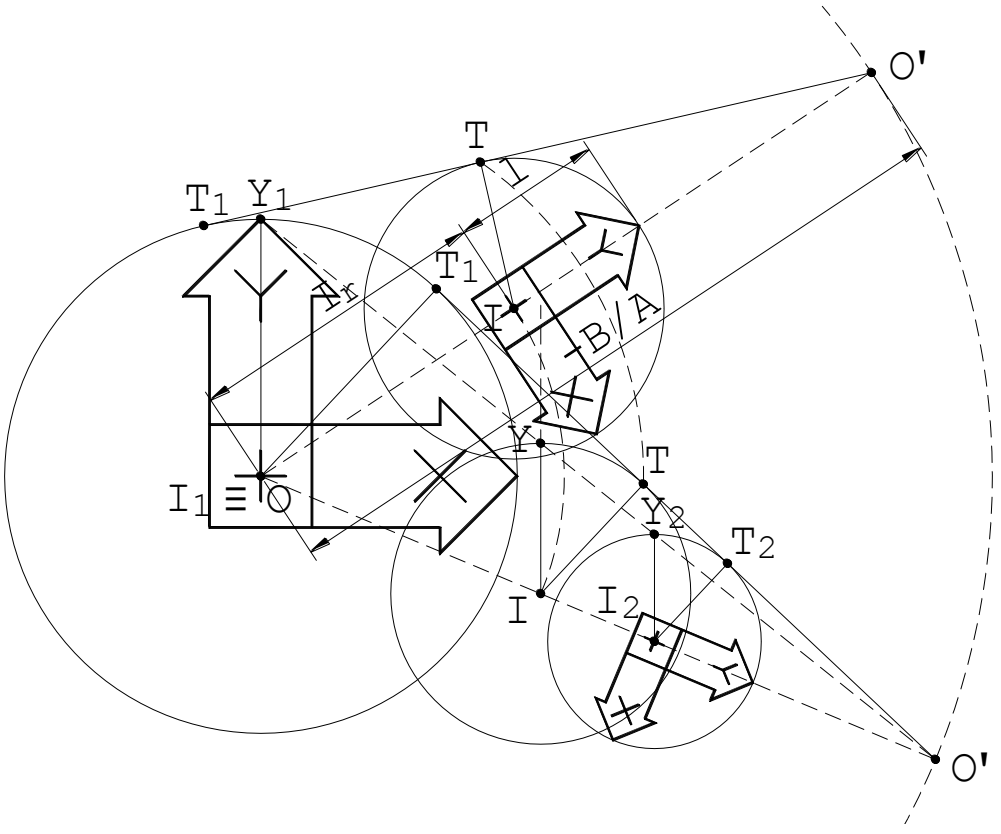


Figura 4.1

Queda per a més endavant la qüestió de quina mida $\mathbf{l_u}$ haurà de fer un bloc que s'insereixi prenent com a factor d'escala la distància complementària $-\left(\mathbf{I_r} + \frac{\mathbf{B}}{\mathbf{A}}\right)$.

De moment, ens permetrem un parèntesi per valorar plantejaments alternatius en relació a la problemàtica del factor d'escala i la dimensió $\mathbf{l_u}$.

Si ens centrem en les dues figures subministrades i comencem limitant-nos a punts **I** alineats amb $\mathbf{I_1}$ i $\mathbf{I_2}$, ideant solucions que després miraríem de generalitzar a d'altres posicions, allò que sens dubte se'ns acudiria primer seria utilitzar com a bloc una figura en què els braços fessin $\mathbf{l_u = 1}$ o, millor, considerar cercles circumscrits a les figures subministrades (per tal que l'orientació no interferís amb la qüestió de l'escala) i adoptar-ne un de radi **1** com a bloc, usant com a factor d'escala la longitud prevista per a la inserció. Tot i manipular cercles substituts, ben segur que en un primer moment prendríem el radi vertical o l'horitzontal com a element de referència (en definitiva, els braços de la figura substituïda), i això ens obligaria a traçar una línia vertical, per exemple, des del punt **I** fins a interceptar la recta $\mathbf{O'–Y_1–Y_2}$, perquè no hi ha cap altra manera d'avaluar la longitud **l**. Però, per poc que seguíssim reflexionant, acabaríem veient que surt més a compte considerar els radis definits pels punts de tangència $\mathbf{T_1}$ i $\mathbf{T_2}$, l'avantatge indiscutible dels quals és que per avaluar distàncies ja no cal dibuixar cap recta abans de cada inserció: un cop tinguéssim $\mathbf{O'–T_2–T_1}$, la longitud $\mathbf{l = I–T}$ la prendríem directament, amb el concurs del mode ***PERpendicular***.

Però el mètode no és extrapolable a punts **I** no alineats amb $\mathbf{I_1}$ i $\mathbf{I_2}$ (Figura 4.1, a la dreta i amunt de $\mathbf{I_1}$): cada vegada caldria copiar la tangent $\mathbf{T_1–T_2}$ i girar-la un angle $\mathbf{I_\varphi – I_{2\varphi}}$, per prendre'n la distància $\mathbf{l = I–T}$. Mirant de reduir les operacions prèvies a cada inserció (l'últim que hem dit representaria executar ***COPIA*** i ***GIRA*** abans de cada ordre ***INSERT***, sense comptar que després hauríem d'esborrar totes les còpies girades de $\mathbf{T_1–T_2}$), també podríem haver pensat a prolongar les línies $\mathbf{I_1–I_2}$ i $\mathbf{T_1–T_2}$ fins a la intersecció $\mathbf{O'}$, i dibuixar el cercle de referència: així, abans d'inserir el bloc a cada nou punt **I** n'hi hagués hagut prou a executar ***LINEA*** per dibuixar una recta des de **I** fins al punt més proper d'aquest cercle (recta que seria radial) i des d'aquí una tangent al cercle substitut centrat a $\mathbf{I_1}$. Arribats aquí, gairebé segur que ens cauria la bena dels ulls i, disposant ja del cercle de referència, ens adonaríem que era encara més senzill usar directament com a factor d'escala la distància mínima $\mathbf{I–O'}$, proporcional a $\mathbf{I–T}$ per a qualsevol posició **I**.

Donem per acabat el parèntesi i seguim. Però abans d'ocupar-nos de la determinació gràfica de l'angle de rotació, encara hi havia un tema pendent: què passa quan $\mathbf{I_{1r} \neq 0 \neq I_{2r}}$? Doncs bé, en aquest cas cal que ens explicitin el centre **O** o bé que ens localitzin a cada figura un punt $\mathbf{Q_i}$ alineat radialment amb el punt d'inserció $\mathbf{I_i}$ (no cal que $\mathbf{Q_1}$ i $\mathbf{Q_2}$ siguin homòlegs) per obtenir el centre com a intersecció de les rectes $\mathbf{I_1–Q_1}$ i $\mathbf{I_2–Q_2}$ (naturalment, això només serà possible si $\mathbf{I_1}$ i $\mathbf{I_2}$ no estan alineats radialment). A més, cal que s'acompleixin quatre condicions:

1) Ha de ser $\mathbf{l_1 \neq l_2}$; altrament, el sistema restaria indeterminat (si $\mathbf{I_{1r} = I_{2r}}$) o representaria el cas trivial en què el coeficient de l'equació és $\mathbf{A = 0}$ i totes les insercions tenen la mateixa grandària $\mathbf{l = B}$ (si $\mathbf{I_{1r} \neq I_{2r}}$), supòsit exclòs dels Exercicis 3 i 4.
2) Com dèiem a l'enunciat, ha de ser $\mathbf{I_{1r} \neq I_{2r}}$; altrament ($\mathbf{l_1 \neq l_2}$ i $\mathbf{I_{1r} = I_{2r}}$), el sistema seria incompatible.
3) Si $\mathbf{I_{ir} < I_{jr}}$, ha de ser $\mathbf{l_i > l_j}$; altrament seríem en el domini de l'Exercici 3.
4) Les figures subministrades han de tenir la mateixa orientació relativa a **O**, requisit que es podria expressar dient que, si un punt $\mathbf{R_1}$ de la primera està alineat amb $\mathbf{O–I_1}$ (ordenació $\mathbf{O–I_1–R_1}$, $\mathbf{O–R_1–I_1}$ o $\mathbf{R_1–O–I_1}$), el seu homòleg $\mathbf{R_2}$ a la segona haurà d'estar alineat amb $\mathbf{O–I_2}$ (mantenint l'ordenació $\mathbf{O–I_2–R_2}$, $\mathbf{O–R_2–I_2}$ o $\mathbf{R_2–O–I_2}$, respectivament). Això només caldria verificar-ho si ens han explicitat el centre **O**; si l'hem deduït de dues alineacions radials $\mathbf{I_1–Q_1}$ i $\mathbf{I_2–Q_2}$, l'acompliment d'aquesta condició hi és implícita.

La determinació del cercle de referència dependrà de si les dues figures estan alineades radialment ($\mathbf{I_{1\varphi} = I_{2\varphi}}$) o no (Figura 4.2): en el primer cas treballarem directament amb les figures subministrades, i en el segon n'haurem de substituir una per una còpia (ATENCIÓ: l'original no ha de desaparèixer) girada al voltant de **O** fins a tenir $\mathbf{I_1}$ i $\mathbf{I_2}$ alineats amb aquest centre i situats a una mateixa banda. A partir d'aquí, localitzarem en les dues figures sengles punts **P**, homòlegs i tals que $\mathbf{P_\varphi \neq I_\varphi}$, i n'obtindrem $\mathbf{O'}$ com a intersecció de les rectes $\mathbf{O–I_1–I_2}$ i $\mathbf{P_1–P_2}$.

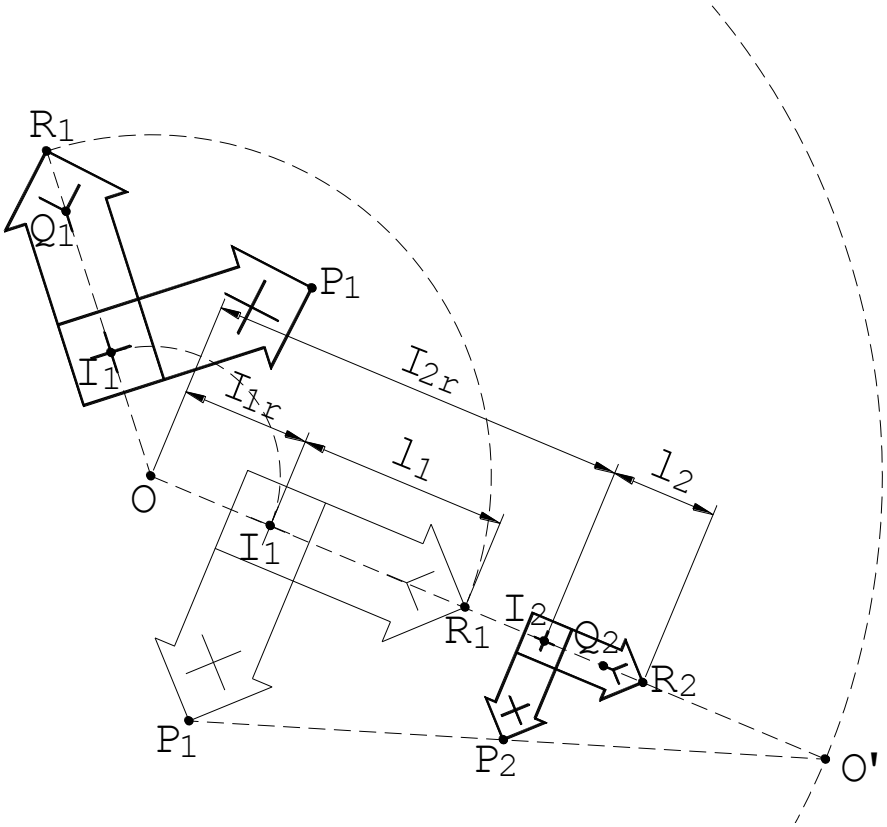


Figura 4.2

Determinació gràfica de l'angle de rotació α

En una inserció a **I**, la figura (per ser més precisos, l'horitzontal de la figura en repòs, que en el nostre cas està representada pel braç o eix **X**) ha de quedar amb una inclinació $\mathbf{I_\varphi - 90^o}$. Si anomenem α l'angle de rotació utilitzat i α_0 la inclinació de la figura que hem adoptat com a bloc, la primera idea que se'ns acut és partir d'una figura en repòs, amb el braç **X** orientat cap a la dreta ($\alpha_0 = 0^o$), i inserir-la amb un angle de rotació $\alpha = \mathbf{I_\varphi - 90^o}$. Però de seguida, quan vulguem posar-la en pràctica, haurem de reconsiderar-la per poc operativa: així com l'orientació centrípeta o radial $\alpha = \mathbf{I_\varphi - 180^o}$ és immediata a partir del punt d'inserció **I** (amb el mode ***PUNto*** de referència a objectes, si **O** està representat per un objecte punt, o amb el mode ***PERpendicular*** aplicat al cercle de referència, ja que el mode ***CENtro*** pot ser conflictiu), per definir l'orientació orbital o tangencial $\alpha = \mathbf{I_\varphi - 90^o}$, normal a l'anterior, hauríem de dibuixar abans una línia **I–P** perpendicular a **I–O** (de fet, la manera més senzilla d'obtenir-la és dibuixar **I–O** o **I–O'** i després girar-la **90°** al voltant de **I**), opció rebutjable sense cap mena de dubte, per la feinada que representa (passaríem de **n** execucions de ***INSERT*** a **n** ***INSERT***, **n** ***LINEA***, **n** ***GIRA*** i una execució de ***BORRA*** per eliminar les **n** línies auxiliars creades i girades). Així que, quan ***INSERT*** ens demani l'angle de rotació li subministrarem $\alpha = \mathbf{I_\varphi - 180^o}$ mitjançant alguns dels dispositius suggerits, i en l'apartat que ve tot seguit ja decidirem amb quina orientació cal definir el bloc partint d'aquesta premissa.

Determinació de la dimensió l_u i l'orientació α_0 del bloc. Creació d'aquest bloc

Si, dibuixat el cercle de referència, escollim una de les figures subministrades, per exemple la inserida a $\mathbf{I_1}$ (millor la més gran, encara que no estigui girada), el factor d'escala corresponent a aquesta inserció serà la distància $\mathbf{I_1–O'}$, que en la expressió

$$\mathbf{l_1 = A\ I_{1r} + B = A \left(I_{1r} + \frac{B}{A} \right) = - A\ d_1}$$

està representada per $\mathbf{d_1 = -\left(I_{1r} + \frac{B}{A}\right) = -\frac{B}{A}}$. Així doncs, la longitud $\mathbf{l_u}$ dels braços de la figura que usem com a bloc (el radi del seu cercle substitut) haurà de ser $\mathbf{l_u = -A = \frac{l_1}{d_1}}$. Com ja hem vist en exercicis precedents, el procediment gràfic més directe per obtenir una figura $\mathbf{d_1}$ vegades més petita que la situada a $\mathbf{I_1}$ és definir-la com a homotètica d'aquesta respecte a $\mathbf{O'}$ i situar-la a una distància **1** d'aquest punt: en la pràctica, copiarem el model en la seva posició original i escalarem la còpia respecte a $\mathbf{O'}$, usant com a factor d'escala $\mathbf{\frac{1}{l_1}}$:

Comando: ***COPIA***
Designe objetos: ***<selecció del conjunt d'objectes que integren la figura>***
Designe objetos:
Precise punto base o de desplazamiento [Múltiple]: ***0,0***
Precise segundo punto del desplazamiento o <usar primero como desplazamiento>:

Comando: ***ESCALA***
Designe objetos: ***P***
Designe objetos:
Precise punto base: ***<clic sobre O'>***
Precise factor de escala o [Referencia]: ***R***
Precise longitud de referencia <1>: ***@***
Designe segundo punto: ***<clic sobre I_1>***
Precise nueva longitud: ***1***

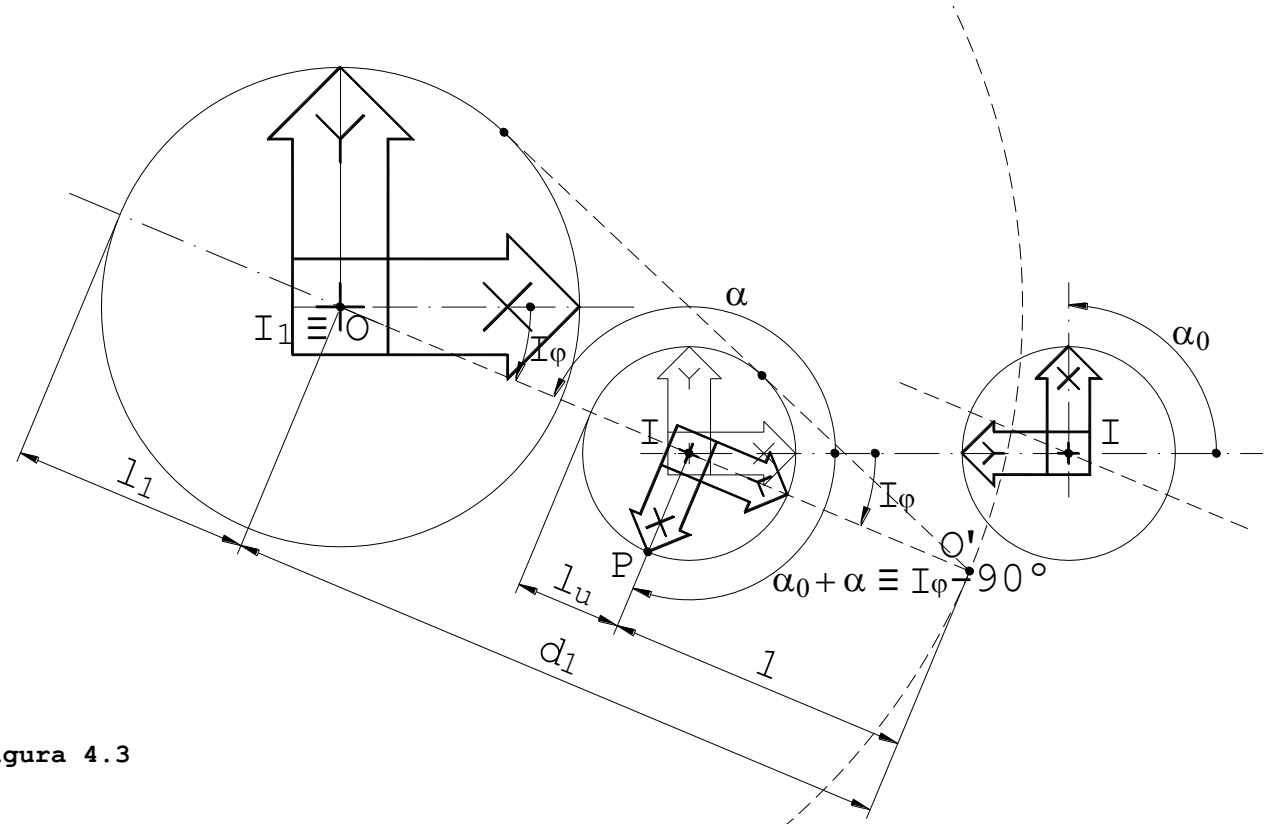


Figura 4.3

Quant a l'orientació $\mathbf{\alpha_0}$ de la figura que utilitzarem com a bloc, partirem del que havíem convingut a l'apartat precedent (inserir-lo amb un angle $\mathbf{\alpha = I_\varphi - 180°}$, que equival a apuntar cap al centre **O**, simplement perquè això era més fàcil de fer) i, tenint en compte que ha d'acomplir-se que $\mathbf{\alpha_0 + \alpha = I_\varphi - 90°}$ (o $\mathbf{\alpha = I_\varphi - 90° - \alpha_0}$, com escrivíem en el quadre inicial), ara ens tocarà decidir entre dues possibilitats:

- Si fem $\mathbf{\alpha_0 = 0°}$, la figura quedarà inserida amb una inclinació $\mathbf{\alpha_0 + \alpha = I_\varphi - 180°}$ (el braç **X** assenyalarà cap al centre **O**) i caldrà donar-li un gir addicional de **90°** al voltant de **I** per tal d'ajustar-lo a $\mathbf{I_\varphi - 90°}$.
- Si fem $\mathbf{\alpha_0 = 90°}$, la figura quedarà inserida amb una inclinació $\mathbf{\alpha_0 + \alpha = I_\varphi - 90°}$ (el braç **Y** adoptarà una orientació centrífuga en relació al centre **O**) i ja no caldrà tocar-la.

Obviament, la segona opció és la correcta, perquè la primera ens obligaria a executar **2 n** ordres (**n *INSERT*** i **n *GIRA***) per deixar les **n** insercions correctament orientades, tot i que això representi un avenç respecte a les **3 n + 1** ordres que hagués comportat inserir segons direccions orbitals (apartat precedent). Així doncs, un cop girada la figura unitària ja podrem fer-ne un bloc:

Comando: ***GIRA***
Angulo actual positivo en SCP: ANGDIR=sentido horario inverso ANGBASE=0
Designe objetos: ***P***
Designe objetos:
Precise punto base: ***<clic sobre* I>**
Precise ángulo de rotación o [Referencia]: ***90***

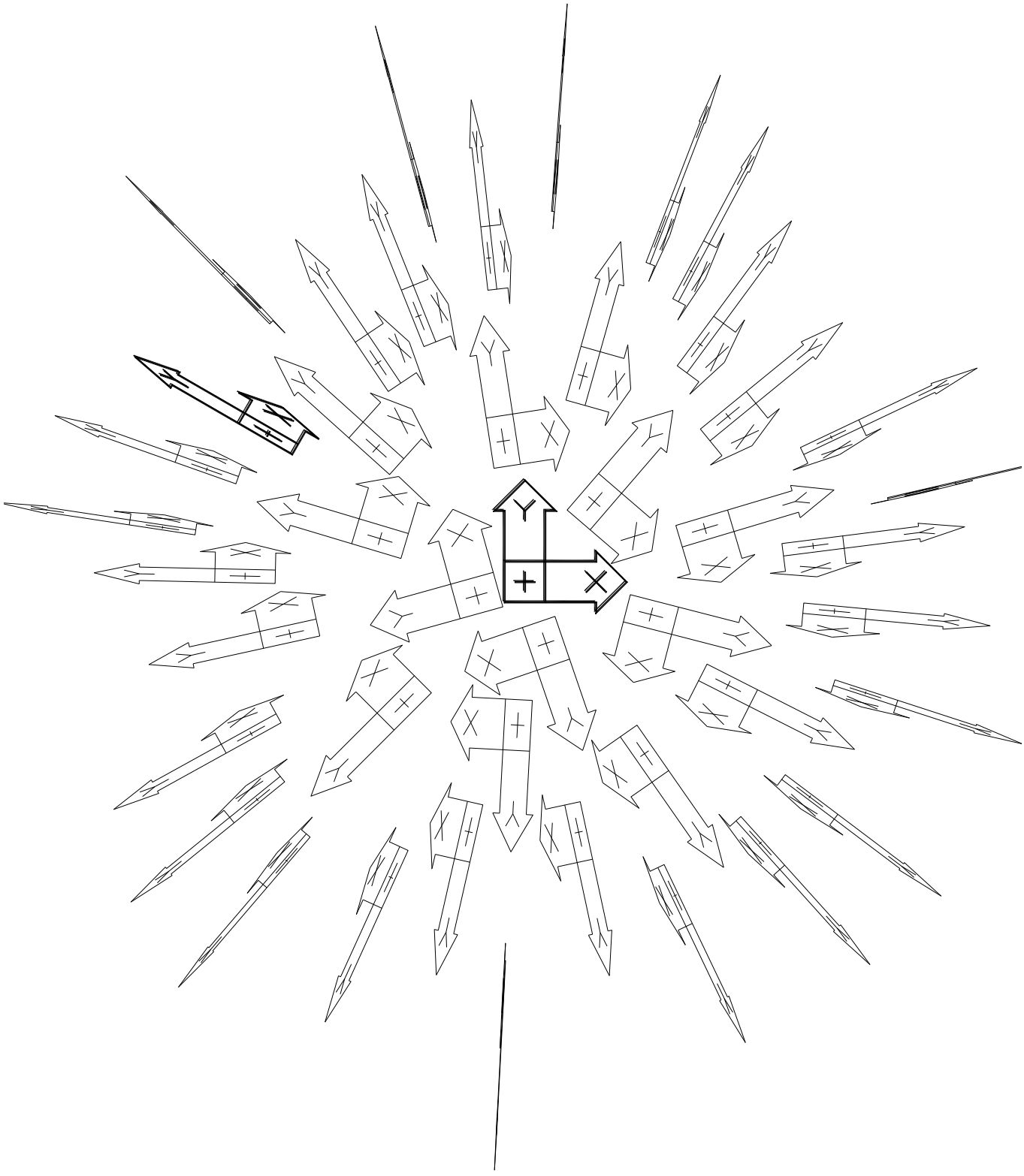
Comando: ***-BLOQUE***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto base de inserción: ***<clic sobre* I>**
Designe objetos: ***P***
Designe objetos:

Inserció del bloc

Com havíem anunciat a la Introducció, cal realitzar les insercions sobre posicions consolidades d'alguna manera. Podem suposar, per exemple, que el mateix client que ens ha encomanat omplir el dibuix amb 500 figures que han de respondre a la llei deduïble de les dues ja subministrades, s'ha encarregat de dibuixar 498 objectes punt (ordre ***PUNTO***) en els llocs on les vol, a més de les 2 figures de referència (per cert que, si un cop dibuixat el cercle de referència, resultés que alguns d'aquests punts se situen per fora, el client s'hauria de pronunciar sobre el particular, dient-nos si cal passar d'aquestes posicions, si cal inserir el bloc com a l'altra banda, sense afectació de l'angle de gir, o bé si cal fer-ho girant-lo 180°). Així doncs, en el guió que clou aquest exercici i que descriu la inserció del bloc a **I** suposarem que en aquesta posició hi tenim un objecte punt, però en consonància amb la notació usada fins ara només escriurem ***<clic sobre* I>**, assumint tàcitament que cal recórrer al mode ***PUNto*** de referència a objectes, i només farem explícita l'aplicació del mode ***PERpendicular*** al cercle de referència.

A la pràctica, abans de procedir a les insercions recorreríem a ***REFENT*** per deixar activats amb caràcter permanent els modes ***PUNto*** i ***PERpendicular***. No n'activarem cap més, perquè, si podem treballar còmodament amb un zoom que permeti la visió íntegra del cercle de referència, l'últim mode ens pot servir tant per al factor d'escala (anant a posicions properes a **I**) com per a l'angle de rotació (anant a posicions allunyades). Altrament, si només volíem tenir en pantalla la regió del cercle de referència immediata a la zona d'operacions, el més intel·ligent hagués estat adoptar com a bloc una figura amb el braç **X** girat cap avall i no cap amunt ($\alpha_0 = -90°$, en comptes de $\alpha_0 = 90°$) i inserir-lo amb orientació centrífuga i no centrípeta (amb un angle $\alpha = I_\varphi$ en comptes de fer-ho amb $\alpha = I_\varphi - 180°$), cosa que ens hauria permès d'aprofitar el mode ***PERpendicular*** tant pel factor d'escala com per l'angle de rotació, fent clic en ambdós casos sobre la regió més pròxima del cercle, tot i que amb les premisses establertes també ens en sortirem: sols caldrà que el centre **O** sigui visible i posar-hi un objecte punt per poder fer clic allà mateix; activar el mode ***CENtro*** només per orientar el bloc no resoldria res ja que, en no poder accedir a la regió més allunyada del cercle, l'acció de ***PERpendicular*** adquiriria prioritat sobre la de ***CENtro*** i la posició referenciada seria **O′** en comptes de **O**.

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***E***
Precise factor de escala para los ejes XYZ: ***<clic sobre* I>**
Designe segundo punto: ***<clic sobre el punt més proper del cercle de referència, amb el mode PERpendicular>***
Precise punto de inserción: ***<clic sobre* I>**
Precise ángulo de rotación <0>: ***<clic sobre el punt més allunyat del cercle de referència, amb el mode PERpendicular>***



EXERCICI 5...

Inserció de blocs, amb orientació constant en relació a un punt **O** (angle de rotació $\alpha = I_\varphi - 90° - \alpha_0$), en què tant les dimensions **x** (perpendiculars a **I–O**) com les dimensions **y** (paral·leles a **I–O**) varien linealment amb I_r ($I_r \geq 0$ i I_φ són les coordenades polars del punt d'inserció **I** relatives al punt **O**).

Algebraicament, això s'expressa amb les equacions

$$x = A\ I_r + B$$
$$y = C\ I_r + D$$

on suposem que **x** decreix i que **y** creix amb la distància I_r a **O** ($A < 0$ i $B > 0$, com en l'Exercici 4, i $C > 0$ com en l'Exercici 3).

Si no ens donen els coeficients **A** i **C** ni les constants **B** i **D**, n'hi haurà prou que ens subministrin les insercions realitzades a I_1 i I_2, sempre que $I_{1r} \neq I_{2r}$: amb les coordenades **r** d'aquests punts i els valors x_1, x_2, y_1 i y_2, formaríem dos sistemes de dues equacions i dues incògnites cadascun, d'on trauríem **A**, **B**, **C** i **D**.

Doncs bé: l'exercici consisteix a definir el problema a partir de dues insercions i resoldre'l, però gràficament i no algebraica. Les cinc qüestions a tractar són: donat **I**, determinació gràfica dels factors d'escala E_x i E_y (aspecte crític 1 de la Introducció) i l'angle de rotació α (aspecte 2); partint d'aquests paràmetres, determinació de les dimensions x_u i y_u (aspecte 4) i orientació α_0 (aspecte 5) del bloc que haurem d'utilitzar, i creació d'aquest bloc; finalment, manera d'executar l'ordre ***INSERT*** (aspecte 6).

Determinació gràfica dels factors d'escala E_x i E_y, i de l'angle de rotació α

Com que, a tots els efectes podem assimilar la dimensió **x** a la grandària **l** de l'Exercici 4 i la dimensió **y** a la de l'Exercici 3 (en què únicament haurem de substituir la denominació del coeficient **A** per **C**, i la de la constant **B** per **D**), en el present exercici prescindirem de moltes de les explicacions i raonaments, i anirem directament a les conclusions. Així, anomenant $d' = -\left(I_r + \frac{B}{A}\right) \geq 0$ i $d'' = I_r + \frac{D}{C}$, ja d'entrada enunciarem les proporcionalitats $x = -A\ d'$ i $y = C\ d''$ que ens permetran de definir dos llocs geomètrics, ambdós circumferències amb centre a **O**: el cercle de referència de les **x**, de radi $-\frac{B}{A}$, que és el lloc geomètric de les insercions en què $x = 0$, representant d' la distància mínima des de qualsevol punt interior **I** (fins a l'extrem del seu radi), i el cercle de referència de les **y**, de radi $\frac{D}{C}$, que és el lloc geomètric de les insercions virtuals en què $y = 0$, representant d'' la distància màxima des de qualsevol punt **I** (fins a la intersecció més allunyada de la seva recta diametral).

Si mireu la Figura 5.1, us adonareu que en l'exercici actual, com en els 3 i 4, I_1 coincideix amb el punt **O** ($I_{1r} = 0$ i $I_{1\varphi}$ resta indeterminat), és a dir, que un dels elements subministrats s'insereix precisament a l'origen de les coordenades polars que estructuren el conjunt, posició per a la qual no està definida l'orientació i que ens ha obligat a assignar-li arbitràriament el valor $\alpha = 0$. Però aquesta inserció té una altra particularitat, i és que $x_1 = y_1$ (circumstància que no s'ha de donar necessàriament, i a propòsit de la qual convidem al lector a plantejar-se algun exercici similar en què $x_1 \neq y_1$). Insistim en això perquè el fet podria ser

la causa d'un lamentable equívoc si, tal i com hem recomanat en els Exercicis 3 i 4 per evitar interferències amb la problemàtica de l'orientació mentre analitzem el tema dels factors d'escala, substituïm aquesta figura per una circumferència (centrada a $\mathbf{I_1}$ i de radi $\mathbf{x_1 = y_1}$) i, en conseqüència, substituïm l'altra figura per una el·lipse (centrada a $\mathbf{I_2}$, amb $\mathbf{x_2}$ de radi mínim i $\mathbf{y_2}$ de radi màxim).

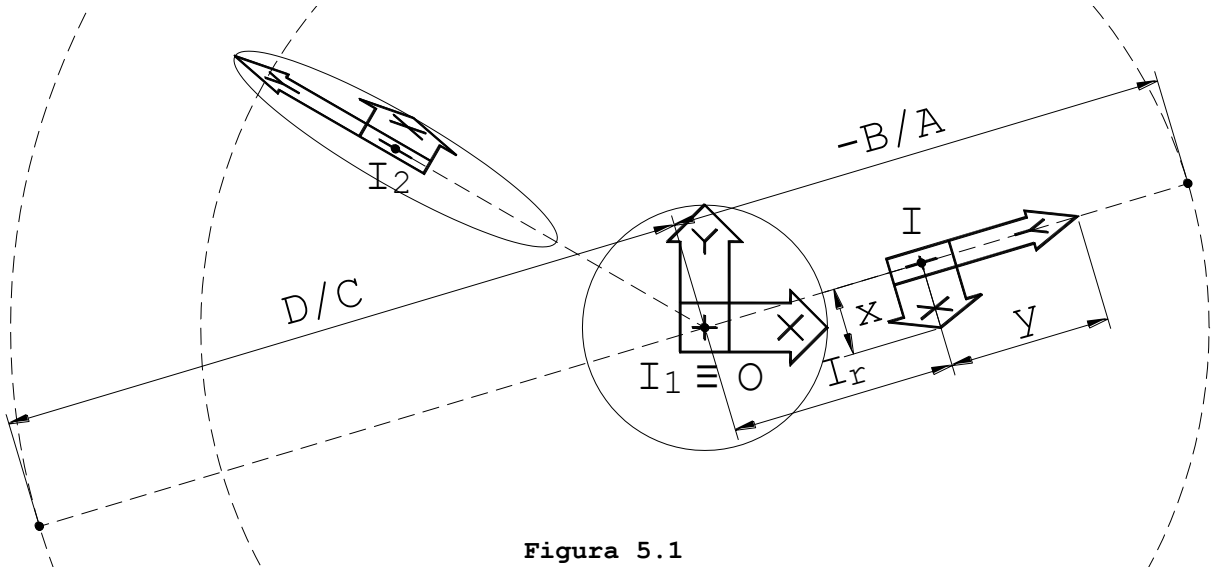


Figura 5.1

La substitució en si no seria perversa (de fet, podríem considerar que la figura no només és el símbol ***SCP*** sinó que la completa un cercle circumscrit, cercle que en qualsevol altra inserció s'hauria transformat en el·lipse), però ho podria ser l'ús que en féssim. Per exemple, si pretenguéssim localitzar un punt del cercle de referència de les **x** fent convergir les dues tangents exteriors a circumferència i el·lipse (facin de substituts o siguin part integrant de les figures, tant se val) cometríem un error, perquè una cosa és considerar que circumferència i el·lipse són figures homològiques en el pla, respecte a un centre $\mathbf{O'''}$ alineat amb $\mathbf{I_1}$ i $\mathbf{I_2}$, i a un eix **e** perpendicular a aquesta alineació (Figura 5.2), correspondència en la qual les referides tangents exteriors són rectes dobles i els punts de tangència $\mathbf{T_1'''}$ i $\mathbf{T_2'''}$ són homològics*, i una altra de molt diferent la correspondència enunciada

---

* Per determinar el centre i l'eix només podríem recórrer a tres parells de punts homològics: els extrems dels diàmetres alineats, $\mathbf{R_1}$ i $\mathbf{R_2}$ més $\mathbf{S_1}$ i $\mathbf{S_2}$, i els punts de tangència $\mathbf{T_1'''}$ i $\mathbf{T_2'''}$ (no podríem comptar amb $\mathbf{I_1}$ i $\mathbf{I_2}$, ni amb $\mathbf{P_1}$ i $\mathbf{P_2}$ o amb $\mathbf{Q_1}$ i $\mathbf{Q_2}$, perquè no són homològics, com veurem). De primer, dibuixaríem la tangent $\mathbf{T_1'''T_2'''}$ i obtindríem el centre d'homologia $\mathbf{O'''}$ per intersecció amb la recta $\mathbf{I_1I_2}$. Pel que fa a l'eix **e**, sols caldria dibuixar una perpendicular a $\mathbf{I_1I_2}$ des del punt d'intersecció de les rectes homològiques $\mathbf{R_1T_1'''}$ i $\mathbf{R_2T_2'''}$ (o de $\mathbf{S_1T_1'''}$ i $\mathbf{S_2T_2'''}$), i ja seria pura rutina obtenir l'homològic de qualsevol punt: en particular, per trobar l'homològic $\mathbf{I_2'''}$ de $\mathbf{I_1}$ uniríem $\mathbf{T_1'''}$ amb $\mathbf{I_1}$ i prolongaríem la recta fins a tallar l'eix **e**; enllaçant aquesta intersecció amb $\mathbf{T_2'''}$ obtindríem $\mathbf{I_2'''}$ i podríem apreciar la no coincidència d'aquest punt amb amb $\mathbf{I_2}$. Val a dir que en aquesta descripció ha quedat un petit detall sense resoldre, tot just començar: el traçat de la tangent exterior a circumferència i el·lipse, que l'ús del mode de referència a objectes ***TANgente*** no resoldrà (de fet, aquesta limitació ens dóna una certa tranquil·litat, perquè pot disuadir els despistats que vagin a ficar-se de peus a la galleda). Com que tampoc hi ha cap mètode gràfic directe que permeti de traçar la tangent, hauríem de recórrer a la determinació numèrica del centre $\mathbf{O'''}$: si anomenem **r** el radi de la circumferència ($\mathbf{r = I_1P_1 = I_1R_1}$), **a** el radi gran de l'el·lipse ($\mathbf{a = I_2R_2}$) i **b** el petit ($\mathbf{b = I_2P_2}$), la distància $\mathbf{O'''_r}$ ($\mathbf{O'''_r = OO'''}$) de $\mathbf{O'''}$ al centre $\mathbf{I_1}$ de la circumferència serà

$$\mathbf{O'''_r = I_{2r} + \frac{I_{2r}^2\, b^2 + r^2\left(a^2 - b^2\right)}{r\sqrt{I_{2r}^2\, b^2 + \left(r^2 - b^2\right)\left(a^2 - b^2\right)} - I_{2r}\, b^2}}$$

on $\mathbf{I_{2r}}$ ($\mathbf{I_{2r} = OI_2}$) és la distància del centre $\mathbf{I_2}$ de l'el·lipse al centre $\mathbf{I_1}$ de la circumferència. Els punts $\mathbf{T_1'''}$ de tangència a la circumferència els podem obtenir gràficament calculant

$$\mathbf{T'''_{1x} = \pm\, r\sqrt{1 - \frac{r^2}{O'''^2_r}}} \qquad \mathbf{T'''_{1y} = \frac{r^2}{O'''_r}}$$

i els punts $\mathbf{T_2'''}$ de tangència a l'el·lipse els podrem precisar fent

$$\mathbf{T'''_{2x} = \pm\, b\sqrt{1 - \frac{a^2}{\left(O'''_r - I_{2r}\right)^2}}} \qquad \mathbf{T'''_{2y} = \frac{a^2}{O'''_r - I_{2r}} + I_{2r}}$$

(Les coordenades cartesianes de $\mathbf{T_1'''}$ i $\mathbf{T_2'''}$ estan referides a un sistema amb l'origen a $\mathbf{O \equiv I_1}$ i l'eix **Y** orientat cap a $\mathbf{I_2}$.)

en el requadre inicial, que restringida a l'alineació $\mathbf{I_1}$–$\mathbf{I_2}$ podríem assimilar a uns doble homotècia: una de centre $\mathbf{O'}$ (alineat amb $\mathbf{I_1}$ i $\mathbf{I_2}$, i situat més enllà del segon punt), de la qual només tindríem en compte les dimensions **x**, i una altra de centre $\mathbf{O''}$ (també alineat amb $\mathbf{I_1}$ i $\mathbf{I_2}$, però situat del primer punt ençà), de la qual només tindríem en compte les dimensions **y**.

En particular, seria fàcil de confondre la primera d'aquestes homotècies amb l'homologia de centre $\mathbf{O'''}$, però per adonar-nos que $\mathbf{O'}$ i $\mathbf{O'''}$ són punts diferents només cal que dibuixem a $\mathbf{I_2}$ un cercle de radi $\mathbf{x_2}$ (figura homotètica de la situada a $\mathbf{I_1}$) i considerem que els punts homotètics de $\mathbf{P_1}$, $\mathbf{I_1}$ i $\mathbf{Q_1}$ són $\mathbf{P_2}$, $\mathbf{I_2}$ i $\mathbf{Q_2}$, mentre que els homològics d'aquests mateixos punts respecte a $\mathbf{O'''}$, $\mathbf{P_2'''}$, $\mathbf{I_2'''}$ i $\mathbf{Q_2'''}$, queden desplaçats cap al centre $\mathbf{O'''}$: en una homologia, la raó simple es manté sobre rectes paral·leles a l'eix **e** (i així, si $\mathbf{P_1}$–$\mathbf{I_1} = \mathbf{I_1}$–$\mathbf{Q_1}$, serà $\mathbf{P_2'''}$–$\mathbf{I_2'''} = \mathbf{I_2'''}$–$\mathbf{Q_2'''}$) però no en altres direccions (així, encara que $\mathbf{R_1}$–$\mathbf{I_1} = \mathbf{I_1}$–$\mathbf{S_1}$, serà $\mathbf{R_2'''}$–$\mathbf{I_2'''} \neq \mathbf{I_2'''}$–$\mathbf{S_2'''}$ i, com que $\mathbf{R_2'''} \equiv \mathbf{R_2}$ i $\mathbf{S_2'''} \equiv \mathbf{S_2}$, $\mathbf{I_2'''} \neq \mathbf{I_2}$ i de retruc $\mathbf{P_2'''} \neq \mathbf{P_2}$ i $\mathbf{Q_2'''} \neq \mathbf{Q_2}$).

Justament, l'ús de circumferències centrades a $\mathbf{I_2}$, homotètiques de la de centre $\mathbf{I_1}$ (substituta o integrant de la figura subministrada en aquest punt), ens farà més senzilla l'obtenció del cercle de referència de les **x** i del de les **y**: una, de radi $\mathbf{x_2} = \mathbf{I_2}$–$\mathbf{P_2}$ (i tangent interior a l'el·lipse), ens permetrà de localitzar un punt $\mathbf{O'}$ del cercle de referència de les **x**, per intersecció de la tangent exterior $\mathbf{T_1'}$–$\mathbf{T_2'}$ amb la recta $\mathbf{I_1}$–$\mathbf{I_2}$; l'altra, de radi $\mathbf{y_2} = \mathbf{I_2}$–$\mathbf{R_2}$ (i tangent exterior a l'el·lipse), ens permetrà de localitzar un punt $\mathbf{O''}$ del cercle de referència de les **y**, per intersecció de la tangent exterior $\mathbf{T_1''}$–$\mathbf{T_2''}$ amb la mateixa recta. I, atès que $\mathbf{x_1} > \mathbf{x_2}$ ($\mathbf{I_1}$–$\mathbf{P_1} > \mathbf{I_2}$–$\mathbf{P_2}$) i que $\mathbf{y_1} < \mathbf{y_2}$ ($\mathbf{I_1}$–$\mathbf{R_1} < \mathbf{I_2}$–$\mathbf{R_2}$), els punts $\mathbf{O'}$ i $\mathbf{O''}$ s'hauran situat a bandes oposades de **O**.

Com hem justificat en els dos exercicis precedents, quan ***INSERT*** ens demani l'angle de rotació li subministrarem $\boldsymbol{\alpha = I_\varphi - 180^\circ}$ (des de tots els punts d'inserció **I** ens orientarem cap al centre **O**) i en l'apartat que ve tot seguit comentarem la posició en què cal definir el bloc partint d'aquesta premissa: en el decurs de l'exposició ens adonarem que les denominacions "cercle de referència de les **x**" i "cercle de referència de les **y**" que acabem d'atribuir a les circumferències de radis **–B/A** i **D/C**, respectivament, només eren provisionals i als efectes d'inserir el bloc cal intercanviar-les.

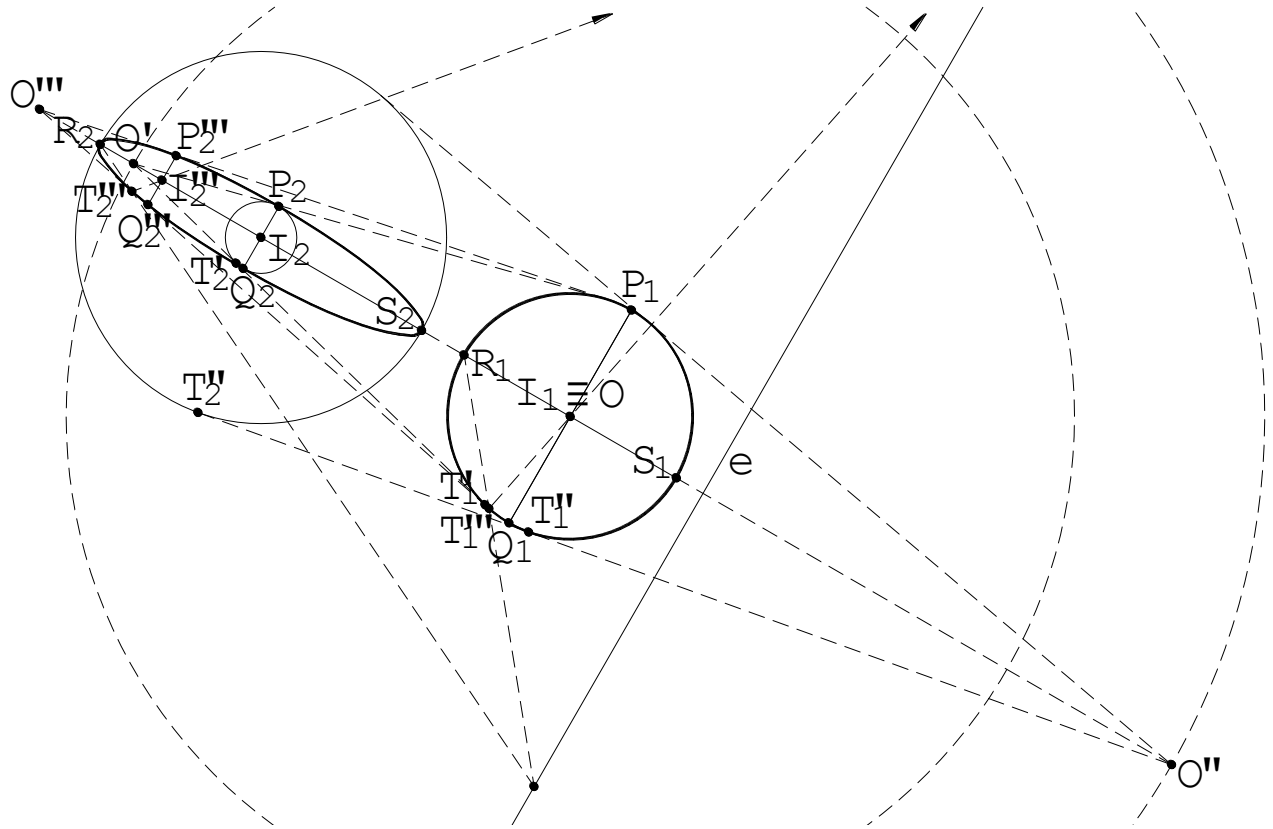


Figura 5.2

Determinació de les dimensións x_u i y_u i l'orientació α_0 del bloc

Un cop dibuixats els cercles de referència, de les dues figures subministrades escollirem la inserida a I_1 (per ser la més gran, però sobretot per ser $x_1 = y_1$). Els factors d'escala corresponents a aquesta inserció seran les distàncies I_1O' i I_1O'', que en en les expressions

$$x_1 = A\ I_{1r} + B = A\left(I_{1r} + \frac{B}{A}\right) = -A\ d_1'$$

$$y_1 = C\ I_{1r} + D = C\left(I_{1r} + \frac{D}{C}\right) = C\ d_1''$$

estan representades per $d_1' = -\left(I_{1r} + \frac{B}{A}\right) = -\frac{B}{A}$ i per $d_1'' = I_{1r} + \frac{D}{C} = \frac{D}{C}$. Així doncs, les longituds x_u i y_u dels braços de la figura que adoptem com a bloc hauran de ser $x_u = -A = \frac{x_1}{d_1'}$ i $y_u = C = \frac{y_1}{d_1''}$. Com ja hem vist en exercicis precedents, el procediment gràfic més directe per obtenir una figura d_1' vegades més petita que la situada a I_1 és definir-la com a homotètica d'aquesta respecte a O' i situar-la a una distància **1** d'aquest punt, i per obtenir-ne una altra d_1' vegades més petita, definir-la com a homotètica respecte a O'' i situar-la a una distància **1** d'aquest punt: en la pràctica, copiarem el model dues vegades en la seva posició original i escalarem cada còpia respecte a O' i O'' respectivament, usant $\frac{1}{x_1}$ i $\frac{1}{y_1}$ com a factors d'escala.

Comando: ***COPIA***
Designe objetos: ***<selecció del cercle substitut centrat a*** I_1 ***>***
Designe objetos:
Precise punto base o de desplazamiento [Múltiple]: ***0,0***
Precise segundo punto del desplazamiento o <usar primero como desplazamiento>:

Comando: ***ESCALA***
Designe objetos: ***P***
Designe objetos:
Precise punto base: ***<clic sobre*** O' ***>***
Precise factor de escala o [Referencia]: ***R***
Precise longitud de referencia <1>: ***@***
Designe segundo punto: ***<clic sobre*** I_1 ***>***
Precise nueva longitud: ***1***

Comando: ***COPIA***
Designe objetos: ***<selecció del cercle substitut centrat a*** I_1 ***>***
Designe objetos:
Precise punto base o de desplazamiento [Múltiple]: ***0,0***
Precise segundo punto del desplazamiento o <usar primero como desplazamiento>:

Comando: ***ESCALA***
Designe objetos: ***P***
Designe objetos:
Precise punto base: ***<clic sobre*** O'' ***>***
Precise factor de escala o [Referencia]: ***R***
Precise longitud de referencia <1>: ***@***
Designe segundo punto: ***<clic sobre*** I_1 ***>***
Precise nueva longitud: ***1***

A diferència dels exercicis 3 i 4, en què només usàvem el cercle substitut per trobar O' i després manipulàvem directament la figura, aquí surt més a compte seguir jugant amb els cercles substituts. Únicament quan ja disposem dels cercles de radi x_u i y_u, els aplegarem en un lloc del dibuix lliure de interferències (punt **U**, a la Figura 5.3) i sobre aquesta plantilla recompondrem la figura per convertir-la en bloc. Tal i com havíem fet a l'Exercici 1, copiarem la inserida a I_1, aprofitant que $x_1 = y_1$, i mitjançant l'ordre ***ESCALA*** (amb el punt d'inserció

com a punt base i usant l'opció ***Referencia*** per determinar indirectament el factor d'escala: volem que la longituds $\mathbf{x_1}$ o $\mathbf{y_1}$ es converteixi en **1**) redimensionarem la còpia de manera que faci $\mathbf{1 \times 1}$ (veieu-la a la Figura 5.3, també centrada a $\mathbf{I_1}$ i més gran que la figura original) i en farem un bloc provisional. Amb el concurs dels modes de referència a objectes ***CENtro*** i ***CUAdrante***, inserirem aquest bloc a **U**, prenent com a factors d'escala $\mathbf{E_x = x_u}$ i $\mathbf{E_y = y_u}$ (longituds dels braços **X** i **Y** a la figura de l'esquerra) però girant-lo un angle $\mathbf{\alpha_0 = 90°}$ (figura de la dreta): així, explosionada amb ***DESCOMP*** aquesta inserció (tot i no ser imprescindible, és recomanable per simplificar la base de dades AutoCAD) i reconvertida en bloc definitiu, en les insercions **I** podrem adoptar un angle de rotació $\mathbf{\alpha = I_\varphi - 180°}$ (orientació centrípeta) i la figura sempre quedarà ben orientada $\mathbf{\alpha_0 + \alpha = I_\varphi - 90°}$ (orbitació en sentit antihorari). Però abans de passar al següent apartat i obrar en conseqüència, cal ser ben conscients de què hem fabricat: un bloc la dimensió **x** del qual és $\mathbf{y_u}$ (això implicarà usar un factor d'escala $\mathbf{E_x = d'' = I\text{–}O'' = I_{1r} + \frac{D}{C}}$) i la **y** és $\mathbf{x_u}$ (això implicarà usar un factor d'escala $\mathbf{E_y = d' = I\text{–}O' = -\left(I_{1r} + \frac{B}{A}\right)}$).

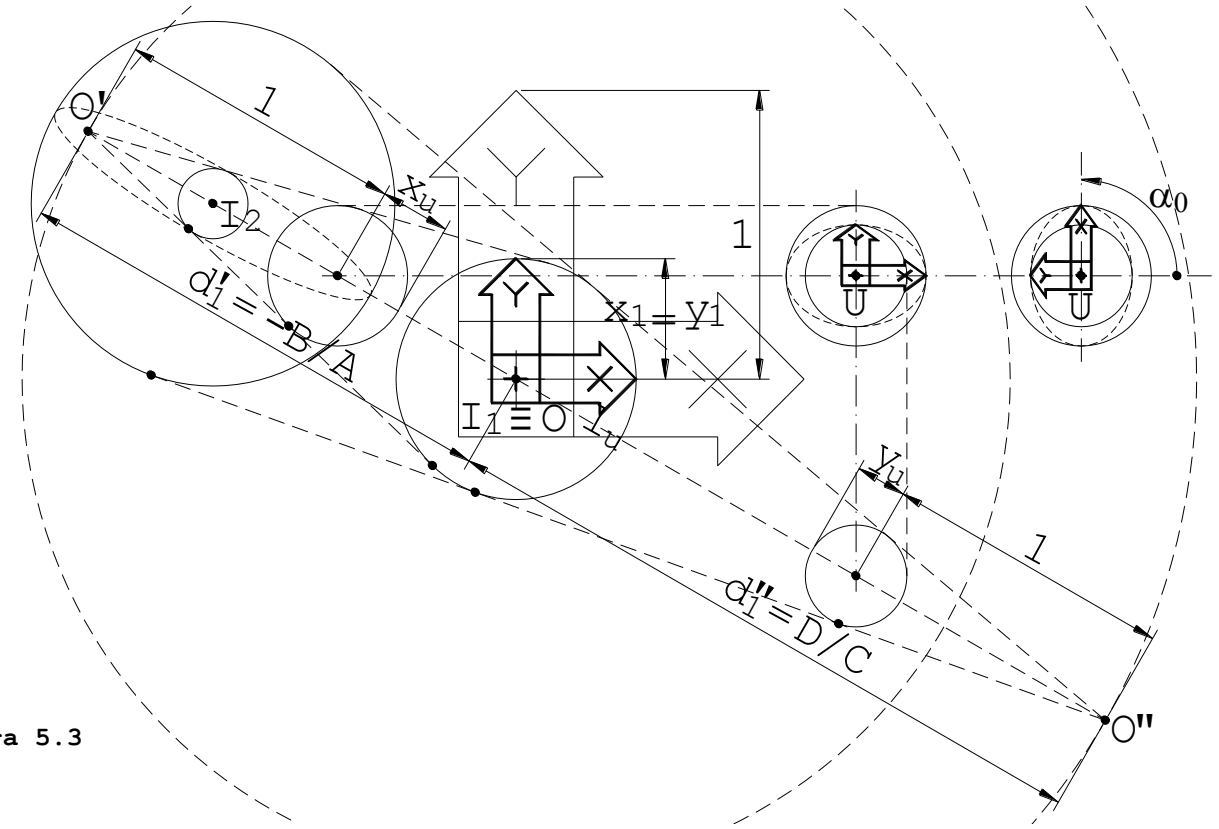


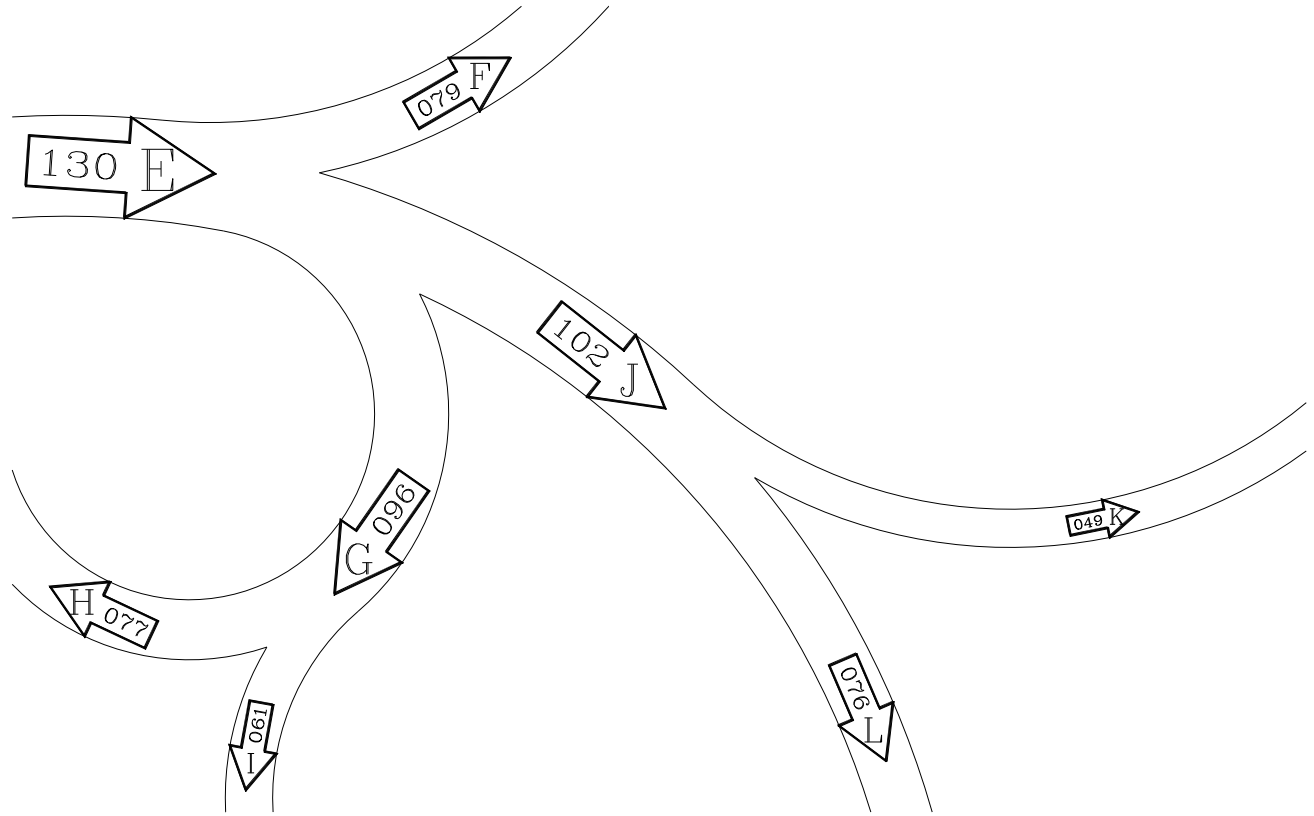
Figura 5.3

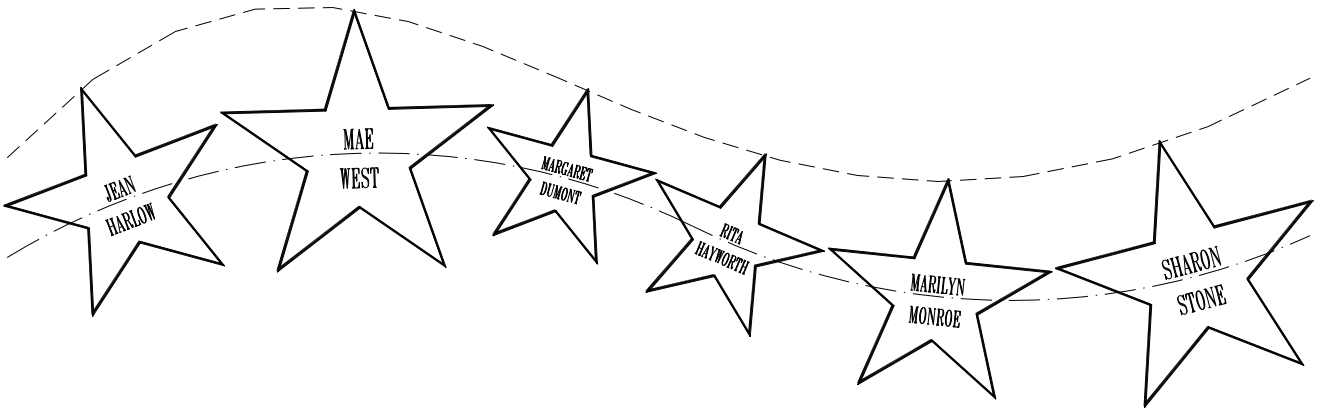
Inserció del bloc

Com havíem anunciat a la Introducció, cal realitzar les insercions sobre posicions consolidades d'alguna manera. Podem suposar, per exemple, que el mateix client que ens ha encomanat omplir el dibuix amb 500 figures que han de respondre a la llei deduïble de les dues ja subministrades, s'ha encarregat de dibuixar 498 objectes punt (ordre ***PUNTO***) en els llocs on les vol, a més de les 2 figures de referència (per cert que, si un cop dibuixat el mal anomenat cercle de referència de les **x**, resultés que alguns d'aquests punts se situen per fora, el client s'hauria de pronunciar sobre el particular, dient-nos si cal passar d'aquestes posicions, si cal inserir el bloc com a l'altra banda, sense afectació de l'angle de gir, o bé si cal fer-ho girant-lo 180°). Així doncs, en el guió que clou aquest exercici i que descriu la inserció del bloc a **I** suposarem que en aquesta posició hi tenim un objecte punt, però en consonància amb la notació usada fins ara només escriurem ***<clic sobre* I>**, assumint tàcitament que cal recórrer al mode ***PUNto*** de referència a objectes, i només farem explícita l'aplicació del mode ***PERpendicular*** als cercles de referència. Quant a la denominació d'aquests cercles, en el guió parlarem de cercle de referència **X** i cercle de referència **Y** d'acord amb el seu ús real.

A la pràctica, abans de procedir a les insercions recorreríem a ***REFENT*** per deixar activats amb caràcter permanent els modes ***PUNto*** i ***PERpendicular***. No n'activarem cap més, perquè aquest últim ens pot servir tant per als factors d'escala com per a l'angle de rotació, anant a buscar la posició del cercle de referència **X** més allunyada de **I** (punt **O″**) per determinar el factor d'escala $\mathbf{E_x}$, la posició del cercle de referència **Y** més propera a **I** (punt **O′**) per determinar el factor d'escala $\mathbf{E_y}$ i la posició més allunyada en qualsevol d'aquests cercles (trobarem abans el de menor radi) per determinar l'angle de rotació **α**: encara que en el centre **O** no hi hagi cap objecte punt (cas en què hi podríem fer clic directament), activar també el mode ***CENtro*** només per orientar el bloc seria una complicació, ja que quan no poséssim massa cura en la determinació del factor d'escala $\mathbf{E_x}$, mirant de no separar-nos massa del punt del cercle de referència **X** diametralment oposat a **I**, l'acció de ***CENtro*** podria adquirir prioritat sobre la de ***PERpendicular*** i la posició referenciada seria **O** en comptes de **O″**.

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***X***
Precise factor de escala X: ***<clic sobre I>***
Designe segundo punto: ***<clic sobre el punt més allunyat del cercle de referència X (O″), amb el mode PERpendicular>***
Precise punto de inserción: ***Y***
Precise factor de escala Y: ***<clic sobre I>***
Designe segundo punto: ***<clic sobre el punt més proper del cercle de referència Y (O′), amb el mode PERpendicular>***
Precise punto de inserción: ***<clic sobre I>***
Precise ángulo de rotación <0>: ***<clic sobre el punt més allunyat d'un dels cercles de referència, amb mode PERpendicular>***


130 E
079 F
102 J
049 K
096 G
H 077
061 I
076 L


JEAN HARLOW
MAE WEST
MARGARET DUMONT
RITA HAYWORTH
MARILYN MONROE
SHARON STONE

... I LA TORNA

Fins aquí hem analitzat casos en què, a partir d'intuicions inicials més o menys encertades, anàvem depurant el plantejament fins assolir un bloc i un procediment d'inserció que garantissin resultats definitius amb el recurs exclusiu a l'ordre ***INSERT***. És cert que de vegades els preparatius (l'obtenció d'elements de suport o referència: eixos coordenats en l'Exercici 1, recta en l'Exercici 2 i cercles en els Exercicis 3, 4 i 5) i la determinació de les direccions i orientació del bloc podien resultar feixucs, però era feina a realitzar una sola vegada i que s'ho valia: n'era la recompensa poder deixar correctament inserides **N** figures amb només **N** ordres ***INSERT*** (si deixem de banda la necessitat de "marcar" prèviament els punts d'inserció, amb ***PUNTO*** o fent ***ID***, en alguns casos).

Però no sempre això és possible: de vegades caldrà dibuixar alguna cosa abans de cada inserció, i de vegades serà després d'inserir el bloc que caldrà ajustar-ne la grandària o l'orientació. El propòsit d'aquest capítol és veure'n algun exemple senzill.

Començarem per un exercici que en les classes d'ELEMENTS DE CAD sempre he proposat (normalment en segon lloc) per il·lustrar l'ús de blocs i atributs, i que segons la classificació de la INTRODUCCIÓ podríem encasellar indistintament en els grups 2.1.1/A (considerant els centres dels parell d'arcs concèntrics) o 2.1.1/B.B (considerant els arcs mateixos). El significat del diagrama (allò que apareix a la Figura 6.1 només n'és un petit fragment, de manera que la utilització de blocs està plenament justificada) és el de menys: un conjunt de pistes o conductes estructurat com un arbre binari que tant podria representar un curs fluvial com els vasos per on circulen fluids orgànics, en un teixit vegetal o animal. L'únic que ens ha d'importar és que cada tram, delimitat per arcs concèntrics, té una amplada constant i que en el sector central ha de dur un símbol en forma de fletxa de proporcions establertes, que incorpora un text numèric situat a la cua i una lletra majúscula centrada en el cap (que a diferència del text numèric, solidari amb la fletxa, sempre es manté horitzontal). A banda d'altres qüestions, tals com caracteritzar i tractar de manera adequada aquests textos i abordar la dualitat dels símbols pel que fa al text de cua (en cinc de les fletxes el nombre s'escriu del darrera cap al davant, mentre que en les tres restants s'escriu del davant al darrera per facilitar-ne la lectura), els reptes principals són: 1) escalar la fletxa de manera que els vèrtexs laterals del triangle cap de fletxa se situïn exactament damunt dels arcs delimitadors del tram, i 2) orientar la fletxa en la direcció i sentit del flux (que els dos punts de contacte esmentats adoptin una posició radial en relació als arcs o, el que és igual, que l'eix de simetria de la fletxa sigui paral·lel a les tangents als arcs en els punts de contacte).

No cal dir que el punt base del bloc haurà de ser un d'aquests dos vèrtexs del triangle (per exemple, l'inferior, referint el qualificatiu a la variant amb major presència), i després dels exercicis comentats tampoc cal entretenir-se massa per justificar les altres dues característiques del bloc: la distància entre els vèrtexs laterals del cap de fletxa (la longitud de la base del triangle isòsceles, equilàter en el cas present) ha de ser **1**, perquè prendrem com a factor d'escala l'amplada de la pista en el tram considerat; la fletxa s'orientarà cap avall (per tal que la base de longitud **1** quedi horitzontal, amb el punt base a l'esquerra), perquè l'angle de rotació el determinarem a partir del punt d'inserció **I** (***CERcano*** sobre un punt de l'arc inferior) en direcció radial, usant ***PERpendicular*** sobre un punt **J** de l'arc superior. La raó d'inserir el bloc amb una orientació radial i no orbital és la mateixa que esgrimíem en els Exercicis 3, 4 i 5: per determinar la primera no cal dibuixar prèviament cap línia perpendicular als arcs concèntrics, mentre que per determinar la segona caldria dibuixar abans de cada inserció una tangent a l'arc inferior (amb l'agreujant que potser n'hauríem de dibuixar més d'una, fins aconseguir que el punt de tangència se situés satisfactòriament cap al mig de l'arc). Quant a la col·locació prèvia d'un objecte punt, per poder-nos referir a la mateixa posició en la determinació del factor d'escala i tot seguit en la introducció del punt d'inserció, a diferència dels Exercicis 2, 3, 4 i 5 aquí no caldrà: no importa que la distància $\mathbf{I_1}\text{¬}\mathbf{J_1}$ a l'arc del davant la prenem des d'una posició $\mathbf{I_1}$ diferent a la del punt d'inserció $\mathbf{I_2}$, perquè al llarg d'un mateix tram de pista l'amplada és constant ($\mathbf{I_1}\text{¬}\mathbf{J_1} = \mathbf{I_2}\text{¬}\mathbf{J_2}$).

En el guió s'emfasitza aquesta circumstància distingint els clics (**I_1** i **I_2** d'una banda, i **J_1** i **J_2** de l'altra) i, a diferència dels exercicis esmentats, en què l'ús del mode ***PUNto*** quedava implícit, aquí s'explicita tant l'ús de ***CERcano*** (la notació ens indica que aquest mode l'activem cada vegada, amb caràcter momentani) com el de ***PERpendicular*** (amb la notació de sempre, que ens recorda la conveniència de recórrer a ***REFENT*** abans de començar les insercions, per deixar activat aquest mode amb caràcter permanent):

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***E***
Precise factor de escala para los ejes XYZ: ***CERcano*** *a* ***<clic sobre I_1>***
Designe segundo punto: ***<clic sobre J_1, amb el mode PERpendicular>***
Precise punto de inserción: ***CERcano*** *a* ***<clic sobre I_2>***
Precise ángulo de rotación <0>: ***<clic sobre J_2, amb el mode PERpendicular>***

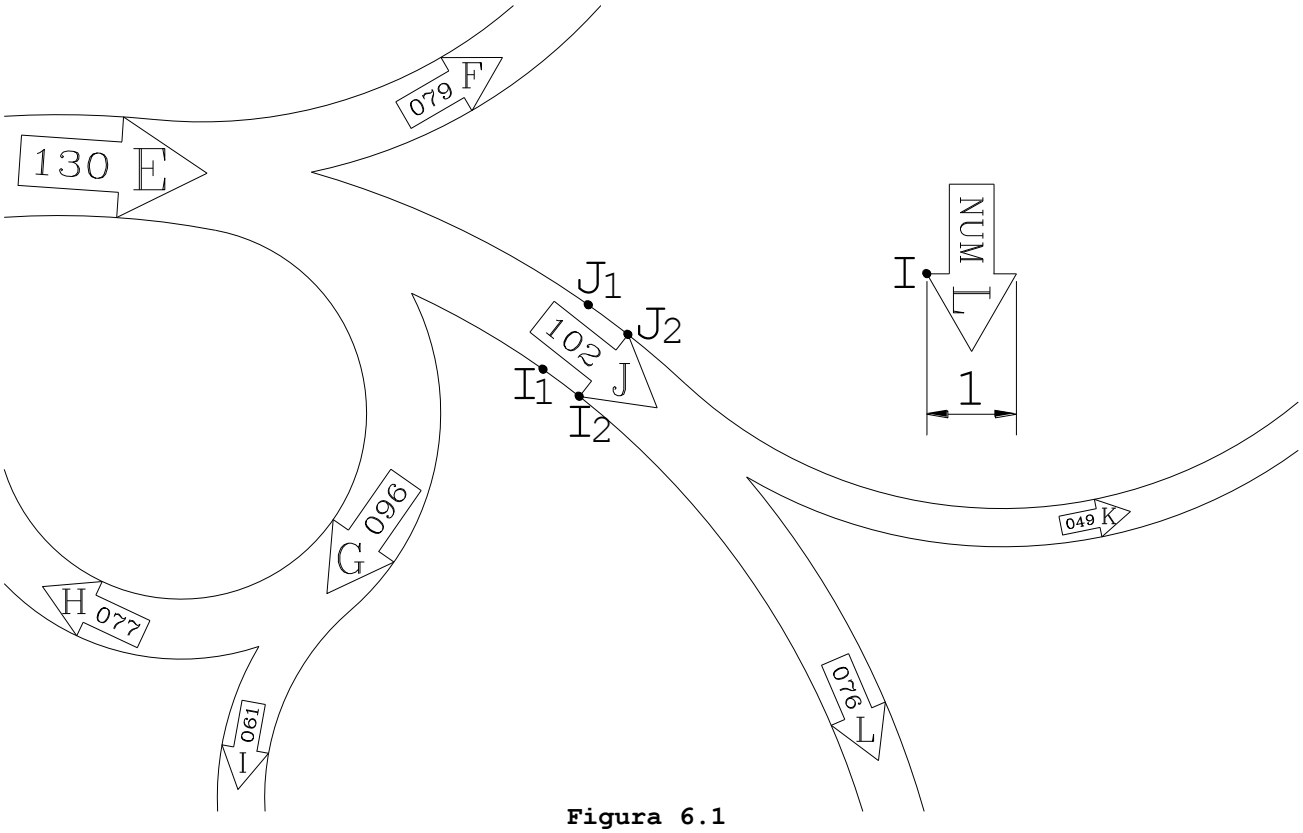


Figura 6.1

Suposem ara que l'aspecte no ens acaba de fer el pes, sobretot en els trams de curvatura pronunciada, pel fet que les fletxes (rectes, llargues i orientades a partir del punt d'inserció) s'acosten massa per la punta i pel darrere a l'arc de major radi. En el cas present, ben segur que la solució passaria per redissenyar el símbol (escurçant la cua o canviant el triangle equilàter per un d'isòsceles rectangle, per exemple) perquè, si provem de centrar la punta de la flecha, la cua encara es desplaçarà més cap a l'arc exterior, i a l'inrevés. Però si, malgrat tot, vulguéssim canviar la segona de les regles del joc, deixant-la així

2a) orientar la fletxa de manera que la punta sigui equidistant dels dos arcs, encara que els dos punts de contacte esmentats no adoptin una posició radial,

o així

2b) orientar la fletxa de manera que l'extrem posterior de la cua equidisti dels dos arcs, encara que els dos punts de contacte no adoptin una posició radial,

la simplicitat de la inserció se n'hauria anat en orris i no hi hauria més remei que realitzar preparatius a cada tram abans d'executar ***INSERT*** (preferentment en una capa diferenciada, per tal d'abordar més còmodament la neteja final). Veiem en què consisteixen, de primer aprofitant el bloc que ja teníem (o, més ben dit, els dos blocs: recordem els dos sentits d'escriptura del text de cua) i tot seguit considerant si amb l'adopció d'un altre bloc (un altre parell) els podem abreujar.

A la Figura 6.2 es representen aquests preparatius: en el centre, un pèl ampliada, una de les fletxes de la Figura 6.1, amb els punts **I_2** i **J_2** d'aquesta figura anomenats ara, simplement, **I** i **J** (per no confondre'ls amb les denominacions **I_1**, **I_2**, **I_3**, **J_1**, **J_2** i **J_3**, que designaran altres punts); a l'esquerra, la inserció de la fletxa a partir de la premissa 2a, i a la dreta, a partir de la premissa 2b.

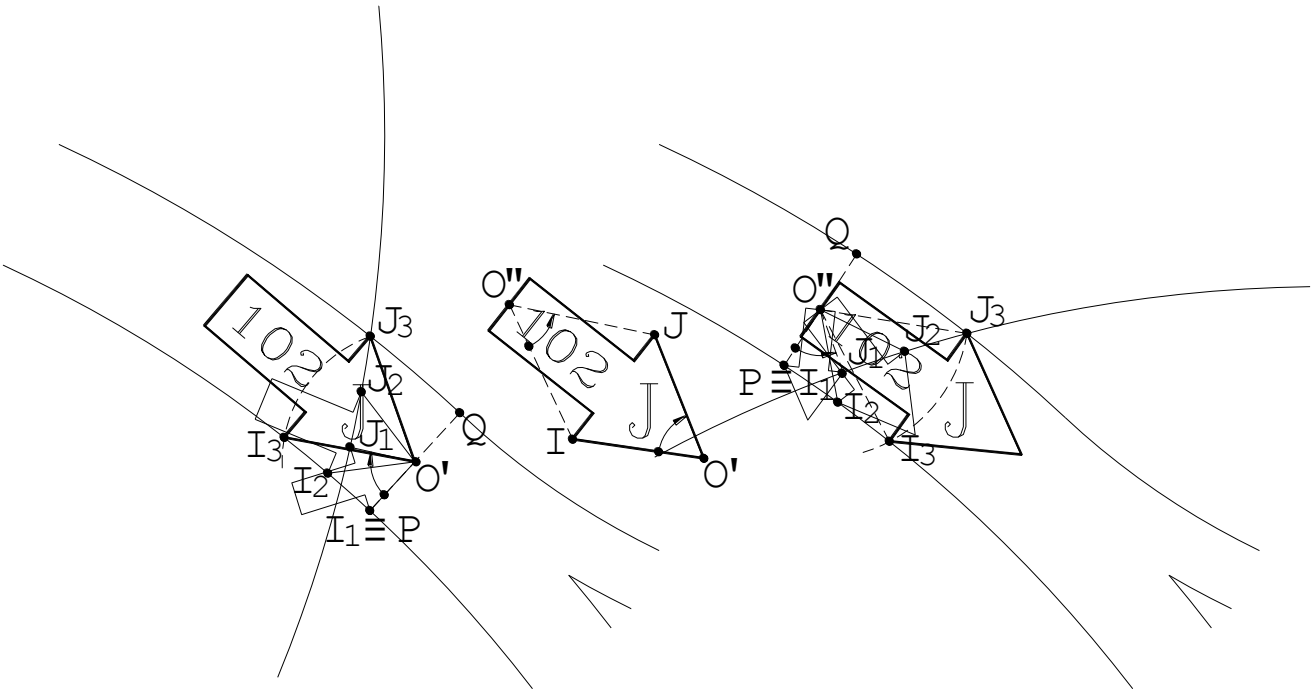


Figura 6.2

La primera cosa que hem de fer és situar el punt **O**, que anomenarem **O′** si correspon a la punta de la fletxa (figura de l'esquerra) o **O″** si correspon al centre de l'extrem posterior de la cua (figura de la dreta): serà el punt ***MEDio*** de la línia **P—Q**, obtinguda des de ***CERcano*** a **P** fins a ***PERpendicular*** a **Q**, si **P** i **Q** pertanyen a arcs concèntrics, o retallant la determinada per ***CENtro*** de **P** i ***CENtro*** de **Q** en cas contrari (si els centres se situessin a una mateixa banda dels arcs, abans caldria allargar-la fins al més exterior). A partir d'aquí, i considerant que **O** és el vèrtex fix d'una sèrie de triangles **OIJ** isòsceles (**O—I = O—J**) i geomètricament semblants (angle **I—O—J** constant), que pertanyen a sengles fletxes (per bé que en el cas **O′** corresponguin al cap i en el cas **O″** no formin part del contorn visible) i que van creixent i girant al voltant de **O** mentre **I** es desplaça sobre l'arc inferior, ens adonem que:

- En una primera posició **I_1 ≡ P**, **O—I_1** serà perpendicular a l'arc inferior i podríem localitzar la posició **J_1** girant **I_1** al voltant de **O** un angle **I—O—J**.
- Per a cada nou contacte **I_2**, ... **I_n** en què **O—I_1 < O—I_2 < ... < O—I_n**, bastiríem la fletxa sobre les noves posicions **J_2**, ... **J_n**, obtingudes girant **I_2**, ... **I_n** al voltant de **O** un angle **I—O—J**, igual com trobàvem **J_1** a partir de **I_1**.
- Observeu que al llarg d'aquesta evolució de les fletxes (girs i escalats respecte al punt **O**), les posicions **I_1**, **I_2**, ... **I_n** del vèrtex **I** descrivien una trajectòria coincident amb l'arc inferior delimitador del tram, i que, en conseqüència, les posicions **J_1**, **J_2**, ... **J_n** (totes elles, obtingudes de les precedents girant un mateix angle **I—O—J**) han de descriure una trajectòria idèntica (un arc d'igual radi) i girada un angle **I—O—J** respecte a **O**.
- Això respon al principi, base de l'instrument anomenat pantògraf, segons el qual el lloc geomètric dels punts **J** que mantenen amb un punt fix **O** i amb posicions mòvils **I** una relació en què tant l'angle **α = I-O-J** com la raó de distàncies **k = O-J / O-I** són constants, és igual a la trajectòria de **I** girada un angle **α** i afectada per un factor d'escala **k** respecte a **O**, aplicable a casos similars.
- L'acompliment de la premissa 1, segons la qual el vértex **J** s'ha de situar sobre l'arc superior delimitador del mateix tram, serà immediata: treiem una còpia de l'arc inferior sobre l'original i girem-la al voltant de **O** un angle **I—O—J** (caldrà usar l'opció **Referencia** de **GIRA** i aplicar-la a qualsevol inserció del bloc, prèviament copiada sobre **O**); la intersecció entre l'arc superior i la còpia girada de l'inferior ens dóna el punt **J** que necessitàvem.

(A la figura hem representat tres posicions de la fletxa: la inicial, que correspon al triangle **OI_1J_1**; una d'intermèdia, que correspon al triangle **OI_2J_2**, i la definitiva, destacada amb traç més gruixut, que correspon al triangle **OI_3J_3**.) Així doncs, un cop realitzats els preparatius en tots els trams executaríem ***REFENT*** per deixar activat amb caràcter permanent el mode ***INTersección*** (ara ja no caldria recórrer a cap altre mode de referència a objectes) i aniríem inserint-hi el bloc tal i com mostra el guió:

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***E***
Precise factor de escala para los ejes XYZ: ***<clic sobre I_3, amb INTersección>***
Designe segundo punto: ***<clic sobre J_3, amb el mode INTersección>***
Precise punto de inserción: ***<clic sobre I_3, amb el mode INTersección>***
Precise ángulo de rotación <0>: ***<clic sobre J_3, amb el mode INTersección>***

I ara ve quan hem de parlar d'optimització: assumit que aquí ja no n'hi ha prou amb ***INSERT***, cal assegurar-se que el nombre d'operacions prèvies a cada inserció és el mínim indispensable. Perquè al lector sagaç no li haurà escapat que en la descripció dels preparatius hem fet una el·lipsi: s'ha parlat de l'obtenció del punt **O**, equidistant dels arcs delimitadors del tram de pista; de treure una còpia de l'arc inferior i de girar-la un angle igual a **I–O–J**, obtenint el punt **J_3**; pero ¿d'on surt **I_3**, del tot necessari com a punt d'inserció que és? Doncs de dibuixar un cercle o un arc amb centre a **O** i que passi per **J_3**, que tallarà just a **I_3** l'arc delimitador inferior. Ara bé: era realment indispensable aquest pas (que, no ho oblidem, haurem de repetir tants cops com insercions haguem de realitzar)? I ara és quan hem de reconèixer que la necessitat de disposar del parell de punts **I_3** i **J_3** només provenia de l'obstinació en aprofitar un bloc pensat per a unes regles de joc diferents (les inicials). Perquè si en volem un que poguem inserir coneixent només un d'aquests punts (**J_3**, per exemple), res més senzill: a diferència del que utilitzàvem, el punt base per a insercions se situaria a **O**, la grandària hauria de ser tal que **O–J = 1** (la nostra fletxa no caldria modificar-la en el cas **O′**, perquè **O–J = I–J = 1**, però en el cas **O″** sí) i l'orientació tal que la línia **O–J** quedés horitzontal, amb el punt base **O** situat a l'esquerra. El guió de la inserció seria:

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***E***
Precise factor de escala para los ejes XYZ: ***<clic sobre O, amb INTersección>***
Designe segundo punto: ***<clic sobre J_3, amb el mode INTersección>***
Precise punto de inserción: ***<clic sobre O, amb el mode INTersección>***
Precise ángulo de rotación <0>: ***<clic sobre J_3, amb el mode INTersección>***

Passarem ara a un exemple on podrem inserir el bloc sense necessitat de preparar el terreny cada vegada, però en què caldrà un ajust *a posteriori* de la grandària de les insercions. Com podeu veure a la Figura 6.3, l'invent podria ser una sanefa o motiu ornamental evocador del "star system" hollywoodenc, en què algunes "sex symbols"*, representades per estrelles de cinc puntes, s'alineen sobre una pauta (una polilínia constituïda per arcs, amb continuïtat a nivell de primera derivada, que anomenarem "de posició"), amb variacions dimensionals donades per una segona pauta (una polilínia, similar a la primera o fruit de l'opció ***curvaB*** de ***EDITPOL***, o bé una corba creada directament amb ***SPLINE***, que anomenarem "de grandària"). Les condicions d'inserció estan prou ben reflectides a la figura: en cada estrella el centre s'ha de situar sobre la pauta de posició, la tangent a la pauta en aquest punt n'ha de marcar l'orientació (els textos hi han de quedar paral·lels) i la punta superior ha d'arribar fins a la pauta de grandària.

* En sentit estricte, atribuir la condició de "sex symbol" a totes les estrelles que figuren a la mostra seria més que discutible. Una només hi té un encaix parcial, perquè els seus personatges, entre paròdics i massa llibertins per a l'època (en el límit del que era assumible per un codi Hays acabat d'estrenar), ho eren de forma explícita, tot i que el seu físic i indumentària responguessin a cànons de bellesa aleshores ja obsolets. L'altra presència atípica, ex-cantant lírica i eterna secundària, té el mèrit d'haver donat la rèplica al més càustic dels còmics de Hollywood en escenes antològiques de sado-masoquisme verbal; entre els cinèfils de pro encara hi ha discrepàncies a l'hora de valorar el seu treball: ¿eren interpretacions magistrals de dona superada per una vertiginosa successió de fets que no acaba d'entendre, o bé l'actriu, atabalada i esmaperduda per l'absurditat del que passava en el plató, al seu voltant, no sabia com reaccionar? Que el lector endevini de quines de les sis estàvem parlant.

Tinguem en compte que, en no ser aquí constant la distància entre ambdues pautes (ni tant sols considerant intèrvals pròxims als punts d'inserció), com sí que ho era la distància entre arcs delimitadors (dintre de cada tram) en l'exercici de les fletxes, no podrem permetre'ns partir d'una posició per avaluar el factor d'escala i després prendre'n una altra com a punt d'inserció: abans de procedir a la inserció d'estrelles, caldrà col·locar sobre la pauta objectes punt per tenir unes referències segures. I depenent de quina sigui la pauta sobre la qual haguem de prefixar les posicions, la resolució serà més simple (només amb ordres ***INSERT***) o més laboriosa (dibuixant perpendiculars a la pauta de posicionsa, abans de fer ***INSERT***, o bé després, ajustant els resultats amb l'ordre ***ESCALA***).

Si ens deixen prendre posicions sobre la pauta de grandària (sanefa superior) la cosa serà senzilla (ni tan sols no necessitarem una tercera pauta auxiliar o de referència): adoptarem com a factor d'escala la distància de ***PUNto*** de **I** fins a ***PERpendicular*** a **J**, inserint a ***PUNto*** de **I** i girant l'angle determinat des d'aquest mateix punt fins a ***PERpendicular*** a ***J***; usarem com a bloc una estrella inscrita en un cercle de radi **1** i amb la punta superior assenyalant a l'esquerra (és a dir, amb els textos girats **90°**), l'extrem de la qual serà el punt base. Però si ens obliguen a partir de punts prèviament situats sobre la pauta de posicions (sanefa inferior), la cosa es complica: haurem de crear una tercera pauta amb ***EQDIST***, rèplica equidistant de la pauta de posicions (la farem per sota de l'original, per no interferir amb la pauta de grandària) només per poder aixecar perpendiculars a la pauta de posicions des de punts **I** de la mateixa pauta i així almenys deixar el bloc ben orientat; quant al factor d'escala (tret que les perpendiculars les haguem materialitzat en línies), com que encara no estem en situació d'adoptar-ne un que garanteixi el contacte de la punta superior amb la pauta de grandària, ens limitarem a inserir provisionalment amb **E = 1** (aquest estadi intermedi apareix a la figura dibuixat amb traç fi). Després d'això, ja estarem en disposició de donar l'últim pas i, mitjançant l'ordre ***ESCALA***, ajustar la mida de l'estrella fins que la punta superior se situï sobre la pauta de grandària. Però per saber-ne la raó de semblança caldrà fer una mica de marxa enrera i parlar del bloc: una estrella també inscrita en un cercle de radi **1** i amb la punta superior assenyalant cap a l'esquerra, però amb el punt base ubicat en el centre, i amb un afegitó que ens permetrà rematar la jugada amb ***ESCALA***. El petit additament només és una línia (millor fer-la en una capa diferenciada) dibuixada des del centre fins força més enllà de la punta superior (podríem usar l'ordre ***RAYO*** en comptes de ***LINEA***, però les semirrectes donen alguns problemes quan formen part de blocs): un cop inserit el bloc amb l'orientació correcta, la intersecció **K** de la línia amb la pauta de grandària ens marcarà la fita que haurà d'assolir la punta superior de l'estrella.

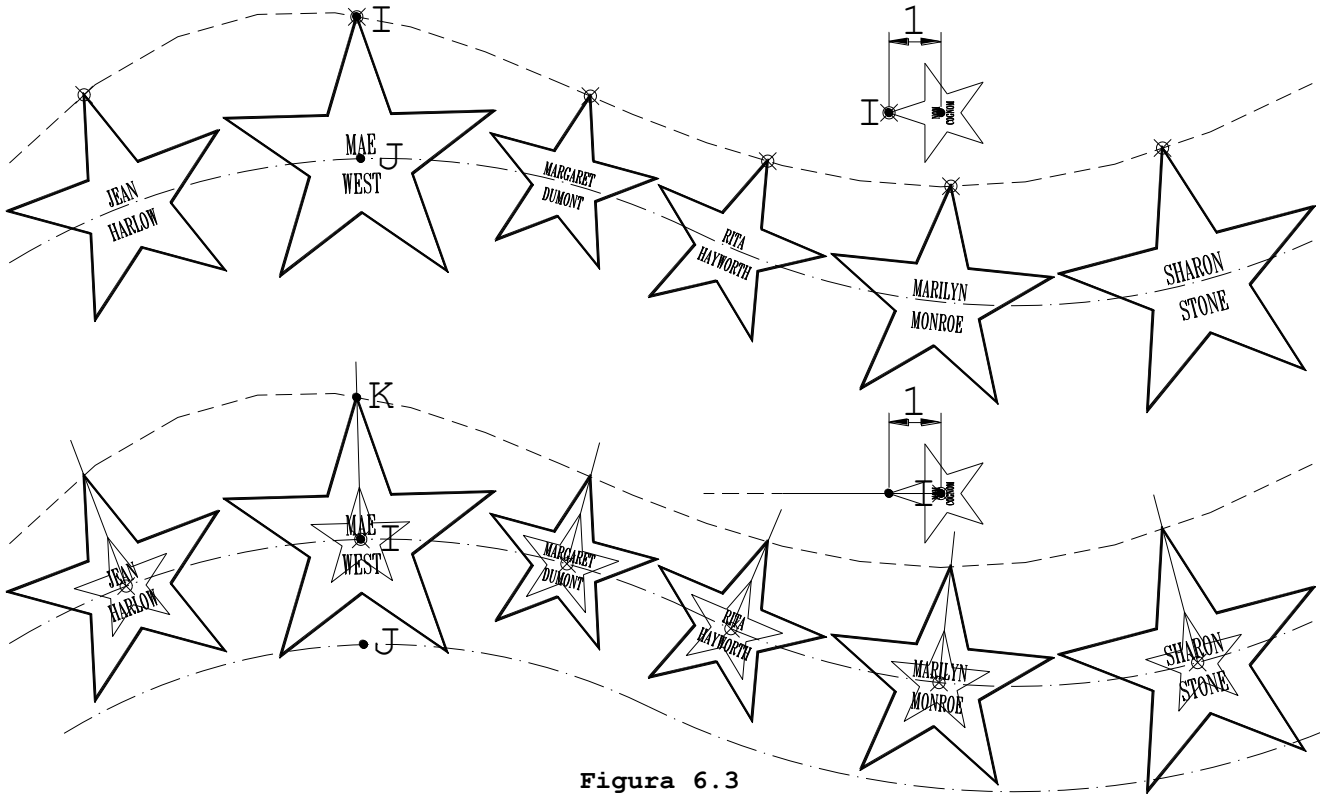


Figura 6.3

En el guió d'una inserció d'aquest segon tipus tornarem a les convencions dels Exercicis 2 al 5, i així, per referir-nos a les posicions prèviament marcades amb objectes punt sols escriurem ***<clic sobre I>***, assumint tàcitament que cal recórrer al mode ***PUNto***, i només explicitarem l'ús de ***PERpendicular*** i ***INTersección*** (a la pràctica, abans d'iniciar les insercions executaríem ***REFENT*** per deixar activats amb caràcter permanent tots tres modes de referència a objectes):

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]:
<clic sobre I>
Indique factor de escala X, precise esquina opuesta, o [Esquina/XYZ] <1>:
Indique factor de escala Y <usar factor de escala X>:
Precise ángulo de rotación <0>: ***<clic sobre J, amb el mode PERpendicular>***

Comando: ***ESCALA***
Designe objetos: ***LT***
Designe objetos:
Precise punto base: ***<clic sobre I>***
Precise factor de escala o [Referencia]: ***R***
Precise longitud de referencia <1>:
Precise nueva longitud: ***<clic sobre K, amb el mode INTersección>***

A més d'un li semblarà, a la vista de l'estratègia seguida, que hem anat de dret a la solució més complicada, perquè l'alternativa que havíem insinuat abans i que respon, com en el primer exercici del capítol, a la modalitat de preparatius previs a la inserció, era més simple: en l'actual, n'hi havia prou a dibuixar una línia des de ***PUNto*** de **I** fins a ***PERpendicular*** a **J**, i allargar-la fins a la pauta de grandària. Amb aquest breu prolegomen, ***INSERT*** podria haver acomplert del tot la seva missió:

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/
Girar/PEscala/PX/PY/PZ/PGirar]: ***E***
Precise factor de escala para los ejes XYZ:
<clic sobre I>
Designe segundo punto: ***<clic sobre K,***
amb el mode INTersección>
Precise punto de inserción: ***<clic sobre I>***
Precise ángulo de rotación <0>: ***<clic sobre J,***
amb el mode PERpendicular>

Ara bé: de què ens serviria aquest mètode amb la variant representada a la Figura 6.4, on la sanefa es desenvolupa de dalt a baix, amb unes estrelles orientades tangencialment a la pauta de posició?. Deixem la interrogació sense resposta, per tal que el lector pugui dir la seva i, després d'atorgar prioritats als diversos condicionaments geomètrics (no s'us passi per alt que el contacte amb la pauta de grandària podria esdevenir-se amb la punta inferior dreta abans que amb la superior dreta, com li hauria pogut passar a l'estrella MAE WEST si la curvatura de la pauta hagués estat més pronunciada), resolgui el problema en conseqüència.

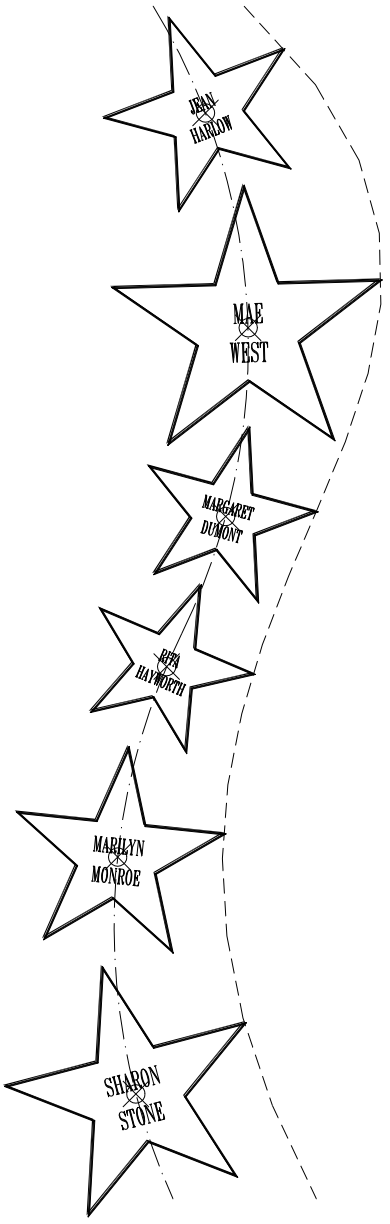
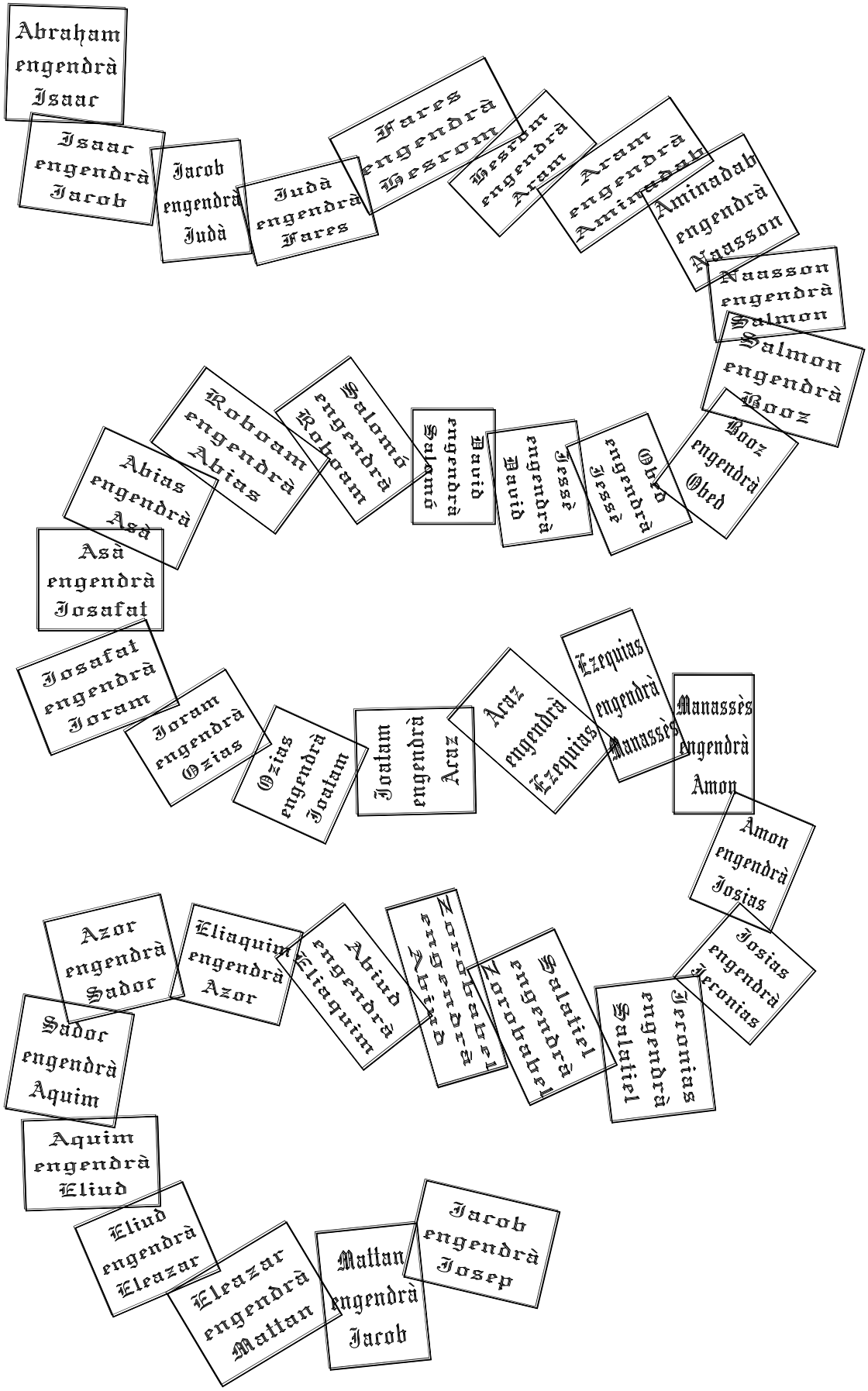


Figura 6.4

Part 2

ESMOLANT LES EINES AMB AUTOLISP


Abraham engendrà Isaac
Isaac engendrà Jacob
Jacob engendrà Judà
Judà engendrà Fares
Fares engendrà Hesrom
Hesrom engendrà Aram
Aram engendrà Aminadab
Aminadab engendrà Naasson
Naasson engendrà Salmon
Salmon engendrà Booz
Booz engendrà Obed
Obed engendrà Jesse
Jesse engendrà David
David engendrà Salomó
Salomó engendrà Roboam
Roboam engendrà Abias
Abias engendrà Asà
Asà engendrà Josafat
Josafat engendrà Joram
Joram engendrà Ozias
Ozias engendrà Joatam
Joatam engendrà Achaz
Achaz engendrà Ezequias
Ezequias engendrà Manassès
Manassès engendrà Amon
Amon engendrà Josias
Josias engendrà Jeconias
Jeconias engendrà Salatiel
Salatiel engendrà Zorobabel
Zorobabel engendrà Abiud
Abiud engendrà Eliaquim
Eliaquim engendrà Azor
Azor engendrà Sadoc
Sadoc engendrà Aquim
Aquim engendrà Eliud
Eliud engendrà Eleazar
Eleazar engendrà Mattan
Mattan engendrà Jacob
Jacob engendrà Josep

OPTIMITZACIÓ DELS RECURSOS EXISTENTS: 2 EXEMPLES

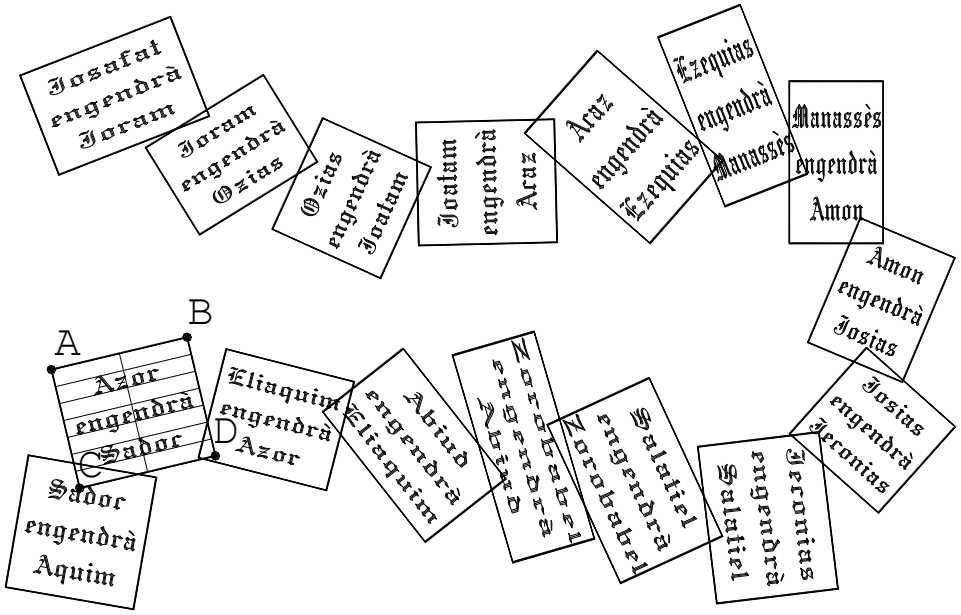
Tot i que el recull d'exercicis no pretenia ser exhaustiu (de fet, només cobreix una petita part del repertori esquematitzat al final de la INTRODUCCIÓ), sí que aspirava a ser representatiu de l'estil de treball que cal adoptar en la resolució de problemes d'aquest tipus, caracteritzat no només pel mètode d'anàlisi sinó per un afany gairebé obssessiu a treure el rendiment màxim de l'ordre ***INSERT***, en el sentit d'aconseguir executar-la amb el mínim nombre d'ínputs. Allò que en un altre context podria semblar innecessari, exagerat o fins i tot ridícul, queda plenament justificat quan la característica principal de la feina encomanada és l'elevada quantitat de vegades que cal repetir un procediment rutinari: això serà titllable de taylorisme pur i dur, però dedicar un esforç suplementari a fer possible que en la inserció d'un bloc el nombre d'ínputs baixi de **N** a **N-1** compensarà sempre que la sèrie d'insercions sobre la qual avaluem l'estalvi sigui prou llarga.

En alguns casos, tanmateix, l'impediment que fa inabastable aquesta fita no prové d'un plantejament erroni o encara poc elaborat sinó de les pròpies limitacions d'unes ordres que, malgrat una acceptable flexibilitat, no s'adapten prou bé a determinades situacions. En aquest capítol en tocarem dues, que farem concórrer en un mateix exemple, i n'oferirem una sol·lució: mitjançant una aplicació AutoLISP, ampliar el repertori AutoCAD incorporant-hi funcions **C:** que complementin les ordres bàsiques o les substitueixin.

Suposem que ens proporcionen un dibuix amb 39 rectangles de diverses proporcions base/altura i també amb orientacions diferents, i ens encomanen d'omplir-los amb 39 dels 40 passos generacionals (l'últim el deixem, per atípic) enumerats en el primer capítol de l'Evangeli segons Mateu, versicles 2 al 16, que tracten de la genealogia de Jesús (amb el masclisme comú a gairebé totes les cultures, ja que només en uns pocs casos es fa esment de les mares) començant per "Abraham engendrà Isaac" i acabant amb "Jacob engendrà Josep", de manera que cada enunciat s'acomodi dintre de la cel·la rectangular que li hagi pertocat, organitzat en tres línies centrades, amb textos i interlineat repartint-se a parts iguals (7 parts) l'altura del rectangle, i amb un factor d'amplada proporcional a la relació base/altura (igual que el tipus de lletra, caldria que ens diguessin quin valor ha de tenir aquest factor d'amplada quan base = altura).

Un cop disposem del bloc adient, un quadrat amb costats de longitud **1**, pautat amb sis línies auxiliars equidistants entre elles i amb les bases, per donar suport a dos objectes ***atributo*** (els dos noms bíblics) i un objecte ***texto*** (el text constant "engendrà"), amb el punt base situat a la cantonada inferior esquerra, el primer obstacle que trobem és que l'ordre ***INSERT*** no dóna les mateixes oportunitats quan l'angle de gir té un valor qualsevol que quan és de **0°**. Si encara no sospiteu on és el problema, només cal que feu una petita prova, dibuixant amb l'ordre ***RECTANG*** un rectangle orientat com el ***SCP*** vigent i inserint-hi un bloc: després d'introduir com a punt d'inserció la cantonada inferior esquerra, n'hi haurà prou a usar el recurs ***Esquina*** i fer clic sobre la cantonada superior dreta (no cal escriure cap ***E***, ja ho sabeu), acabant amb l'acceptació de l'angle de gir per defecte, **0°**; veieu com l'adopció d'un bloc unitari ens ha permès de definir còmodament d'un sol tret els dos factors d'escala, que coincideixen amb la base i l'altura del rectangle. Però quan intenteu repetir la jugada, *mutatis mutandis*, sobre un qualsevol dels rectangles inclinats, ja no tot seran flors i violes: postser esperàveu que en forçar "l'ordre natural" d'introducció de les dades, fent ***Girar*** i introduint els extrems esquerre i dret de la base del rectangle per determinar l'angle de gir abans de subministrar el punt d'inserció, tot aniria igual, però no ha estat així; immediatament després d'haver fet coincidir el punt d'inserció amb la cantonada inferior esquerra sembla que les coses vagin bé, ja que la presentació provisional de resultats que arrossega el cursor és un rectangle amb la inclinació desitjada, però en portar-lo cap a la cantonada superior dreta del rectangle d'acollida veurem que la del segon rectangle no hi convergeix. Si seguim movent el cursor i creuem l'horitzontal o la vertical del punt d'inserció, ens adonarem de què està fallant: crèiem que, un cop desvetllada la incògnita de la inclinació del bloc, ***INSERT*** adaptaria el recurs ***Esquina*** a aquesta circumstància, fent que els factors d'escala $\mathbf{E_x}$ i $\mathbf{E_y}$ es referissin a l'orientació d'un ***SCP*** girat el mateix angle, però potser era demanar massa pretendre que AutoCAD ens endevinés les intencions.

Figura 7.1



De fet, ***Esquina*** sempre juga amb les direccions coordenades, tant és que introduïm l'angle de gir abans com després, i si volem sortir-nos-en no hi haurà més remei que cancel·lar l'ordre ***INSERT*** en curs i tornar-la a executar d'acord amb el guió que presentem (els punts fan referència a un dels rectangles de la Figura 7.1):

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***G***
Precise ángulo de rotación <0>: ***<clic sobre C>***
Designe segundo punto: ***<clic sobre B>***
Precise punto de inserción: ***X***
Precise factor de escala X: ***<clic sobre C>***
Designe segundo punto: ***<clic sobre D>***
Precise punto de inserción: ***Y***
Precise factor de escala Y: ***<clic sobre C>***
Designe segundo punto: ***<clic sobre A>***
Precise punto de inserción: ***<clic sobre C>***
.....

En total (comptant-hi el nom del bloc però ignorant la determinació de valor dels atributs, ja que aquesta és una altra història), 11 ínputs, reduïbles a 9 si ens adonem que ja no serveix de res entrar l'angle de gir en primer lloc (en no poder recórrer a ***Esquina*** per a la determinació gràfica simultània de $\mathbf{E_x}$ i $\mathbf{E_y}$) i fem:

Comando: ***-INSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***X***
Precise factor de escala X: ***<clic sobre C>***
Designe segundo punto: ***<clic sobre D>***
Precise punto de inserción: ***Y***
Precise factor de escala Y: ***<clic sobre C>***
Designe segundo punto: ***<clic sobre A>***
Precise punto de inserción: ***<clic sobre C>***
Precise ángulo de rotación <0>: ***<clic sobre D>***
.....

Així doncs, si la "intel·ligència" d'AutoCAD és tan limitada, haurem d'espavilar-nos pel nostre compte i construir una ordre alternativa capaç de reaccionar com esperàvem que ho fes ***INSERT***, és a dir, funcionant amb només 6 ínputs:

Comando: ***GINSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***G***
Precise ángulo de rotación <0>: ***<clic sobre C>***
Designe segundo punto: ***<clic sobre D>***
Precise punto de inserción: ***<clic sobre C>***
Indique factor de escala X, precise esquina opuesta, o [Esquina/XYZ] <1>:
<clic sobre B>
.....

Fent-la homònima, la nova ordre podria substituir ***INSERT***: definiríem una funció **C:INSERT** i, des del mateix arxiu .LSP on situéssim aquesta definició, executaríem l'ordre ***ANULADEF*** per posar fora de servei l'ordre original i deixar el camp lliure a la substituta. Tanmateix, preferim donar-li una existència palal·lela (no podem dir "independent", perquè mitjançant la funció predefinida **command** seguirem usant ***INSERT***) i anomenar-la ***GINSERT***: inserció amb la ***G*** primer de tot.

Vista una de les limitacions intrínseques objecte d'aquest capítol i anunciada la seva superació (més endavant presentarem **C:GINSERT** i en justificarem el disseny), parlem ara de l'altra, que no té tant a veure amb el nombre d'ínputs de ***INSERT*** com amb la seva longitud (en concret, la dels textos que assignem als atributs del bloc inserit, assignació que en els guions precedents havíem despatxat amb punts suspensius) i amb la possibilitat de reduir-la.

En realitat, escurçar els atributs, així sense més, no seria cap millora sinó una bajanada absoluta, perquè el valor de cada atribut ha de ser aquell que li convé i prou: "Abraham", "Isaac", ... "Jacob" i "Josep", en el nostre cas. El que passa aquí és que aquest escurçament forma part d'una estratègia indirecta per fer front a la murga que seria haver d'escriure dues vegades cada text: dels $39 \times 2 = 78$ valors a assignar, tret del primer i l'últim ("Abraham" i "Josep"), els altres 76 són en realitat 38 textos (de "Isaac" a "Jacob"*) que es repeteixen al llarg de les 39 insercions, repeticions que seria bo de poder-li estalviar a l'usuari. No costaria massa de modificar **C:GINSERT** per tal que, a partir de la segona inserció, una resposta nul·la al requeriment de valor per al primer atribut (nom del pare) comportés l'assignació automàtica de l'últim valor introduït, és a dir, del valor assignat al segon atribut (nom del fill) en la inserció precedent. Sens dubte, aquesta seria la manera directa d'atacar el problema, però obligaria a efectuar les insercions en un ordre precís i la seva utilitat es limitaria a una categoria molt reduïda de casos: aquells en què, com a l'exemple, les insercions defineixen una seqüència lineal on dos dels atributs de cada bloc n'asseguren l'encadenament. Si anem més enllà i ens plantegem alleujar l'usuari en situacions encara pitjors, com quan, en el conjunt de les insercions, els valors assignats als atributs es repeteixen no una sinó unes quantes vegades (sense moure'ns de l'àmbit de les genealogies, pensem en la diagramatització d'un arbre que representés tota la descendència d'un individu fins a una generació concreta i no sols l'ascendència per via masculina d'un dels descendents, on l'estructura no estigués representada només pels enllaços gràfics entre blocs sinó per la informació textual d'aquests i es pogués recórrer en els dos sentits), sense condicionaments pel que fa a l'ordre amb què realitzem les insercions, anirem a parar a l'estratègia següent:

- En l'ordre ***INSERT*** no assignaríem als atributs de cada bloc el valor definitiu sinó un altre de més curt. En el nostre exemple, escriuríem "Abraham" i "01" en la primera inserció, "01" i "02" en la segona, ... "37" i "38" en la penúltima, i "38" i "Josep" en l'última.

- Un cop realitzades totes les insercions, amb l'ordre ***-ATREDIT*** procediríem a la substitució dels valors provisionals pels definitius. Així, substituiríem tots els valors "01" per "Isaac", tots els valors "02" per "Jacob", ... tots els valors "37" per "Mattan" i tots els valors "38" per "Jacob". (Observeu que l'existència d'homònims, com els dos "Jacob" de la 3ª i la 39ª generació, no invaliden el mètode: no hi fa res que un mateix nom sigui temporalment suplantat per més d'un de provisional; l'error vindria de la falta d'univocitat en sentit contrari, és a dir, de que s'assignés un mateix valor provisional a atributs que han de tenir valors diferents o, com és lògic, d'usar com a valors provisionals textos que ja figuressin entre els definitius.)

Quant a l'eficiència del mètode, caldrà tenir present:
- Quants atributs es repeteixen.
- Quantes vegades es repeteixen, de mitjana.
- Nombre de caràcters dels valors provisionals (més un, per l'<Intro>).
- Nombre de caràcters, de mitjana, dels valors definitius (més un, per l'<Intro>).

---

* Els impius i els missaires no tindreu cap problema, però els que conegueu les Escriptures només pel damunt us podríeu fer un embolic de ca l'ample: aclarirem que, entre els avantpassats de Jesús, n'hi ha dos anomenats Jacob: el primer (3ª generació des d'Abraham) era fill d'Isaac i el segon (39ª generació) era el pare de Josep (del Josep pare putatiu de Jesús, perquè amb això podria haver-hi un nou motiu de confusió, derivat del fet que hi ha un altre Josep, germà de Judà i de deu més, fill del primer Jacob).

Si ho reduïssim tot a un simple còmput de pulsacions (la realitat és més complexa, perquè hi intervenen d'altres factors), a l'exemple tindríem, deixant de banda "Abraham" i "Josep":
- 38 atributs que es repeteixen.
- Repetició simple, que comporta $(1 + 1) \times 38 = 76$ atributs que caldria escriure.
- Els valors provisionals tenen 2 caràcters $(2 + 1)$.
- Entre "Asà" (3) i "Zorobabel" (9), la longitud mitjana dels valors definitius és de 5,78 caràcters $(5{,}78 + 1)$.

Si no introduïm cap artifici i l'usuari es limita a inserir els blocs amb els valors d'atribut definitius, haurà efectuat $76 \times 6{,}78 = 515{,}28$ pulsacions.
Si segueix el mètode proposat, haurà efectuat $76 \times 3 = 228$ pulsacions executant ***INSERT*** i $38 \times (3 + 6{,}78) = 371{,}64$ més executant ***-ATREDIT***, amb un total de 599,64 .

El fet que en aquest exemple els comptes no acabin de sortir és el de menys: sempre podríem dir que, si Jesús hagués nascut dos segles més tard, el balanç ens seria més favorable (tot i que, posats a pensar ucronies, probablement la història hauria anat per altres viaranys i ara no ens dedicaríem a aquestes especulacions); en qualsevol cas, sembla indiscutible que quants més elements i/o més repeticions, més justificada estaria l'aplicació d'aquest procediment. I, si alguna cosa falla, no és pas l'estratègia sinó els instruments utilitzats per dur-la a terme. En concret, havíem donat per suposat que l'ordre ***-ATREDIT*** (que és com des d'ACAD 2000 s'anomena l'antiga ***ATREDIT***, mentre que ara aquest nom correspon a l'antiga ***DDATTE***) ens permetria de realitzar totes les substitucions en una única execució i no és aixi, perquè una de dues:

- Si contestem que ***sí*** a la pregunta *¿Editar atributos uno a uno?*, no podrem triar l'ordre amb què AutoCAD ens va presentant els atributs a modificar (dependrà de l'ordre amb què s'hagin fet les insercions, de la variable ***SORTENTS*** i del mètode de selecció), haurem de reclamar l'opció ***Valor*** i la subopció ***Reemplazar*** en cada atribut (són molts els aspectes modificables d'un atribut) i, sobretot, farem un mal negoci perquè l'adopció d'aquestes últimes i l'entrada del nou text les haurem de repetir tantes vegades com aparegui el text provisional a substituir.

- Si contestem que ***no*** i tot seguit donem la mateixa resposta a la pregunta *¿Editar sólo atributos visibles en pantalla?*, podrem especificar directament quin text volem canviar i quin text l'ha de substituir, i a diferència del cas precedent el canvi s'aplicarà automàticament a tots els atributs amb el primer valor. La llàstima és que per cada text provisional a substituir haurem d'executar de nou ***-ATREDIT*** amb tot el que això comporta (tornar a contestar ***no*** a les dues primeres preguntes i a passar un triple filtre de selecció). I si executem ***MULTIPLE*** abans de ***-ATREDIT***, l'únic que estalviarem serà la reiterada invocació d'aquesta ordre.

Ja es veu que la resposta ***no*** (edició global d'atributs) ens dóna la pauta d'allò que buscàvem, i només caldrà posar-li un motor per tal que, havent definit ja d'entrada el camp d'actuació (el conjunt d'atributs susceptible de modificació), AutoCAD permeti introduir a discreció parells de textos <valor_a_substituir> i <valor_susbstitut>. Així doncs, la missió bàsica de l'ordre alternativa que volem crear i que anomenarem ***RATREDIT*** (***R****epetir* ***-ATREDIT***) és "motoritzar" ***-ATREDIT No***. Però, igual que interessa que ***GINSERT*** mantingui totes les possibilitats d'ús de ***INSERT***, fora de quan reclamem l'opció ***Girar*** immediatament després d'haver entrat el nom del bloc (única situació en què ***GINSERT*** tindrà un comportament específic), aniria bé que ***RATREDIT*** donés satisfacció a totes les modalitats d'execució de l'ordre ***-ATREDIT***, incloent-hi la que resulta d'una resposta afirmativa a *¿Editar atributos uno a uno?*, tot i que la motorització només afectés al procés obert per una resposta negativa. I,posats a fer, deixarem que aquesta dinàmica abasti les dues subopcions en què la pregunta *¿Editar sólo atributos visibles en pantalla?* subdivideix aquest procés, per bé que en casos com el de l'exemple la manera més expeditiva d'actuar seria contestar que ***no*** i estendre la selecció a tot el dibuix (fins i tot a les àrees situades visualment fora de camp, als atributs invisibles i als residents en capes desactivades). Naturalment, com que cal decidir què es fa per dir prou (cancel·lar ***RATREDIT*** amb la tecla <Esc> és una solució poc elegant), hem optat perquè, quan <valor_a_substituir> = <valor_substitut>, AutoCAD entengui que ja no volem modificar més atributs. Per no fer el missatge massa llarg, ens limitarem a recomanar que l'usuari pulsi <Intro> en les dues ocasions (tot i que aquesta resposta provocaria en ***-ATREDIT*** l'aparició de l'avís **No válido**, ja ens ho farem perquè això no passi a ***RATREDIT***).

Abans d'entrar en l'anàlisi de les funcions AutoLISP **C:GINSERT** i **C:RATREDIT** que hem preparat per donar satisfacció a les necessitats expressades fins aquí, seria bo completar la descripció de com ens ho faríem per acomplir l'encàrrec amb l'ajut d'aquestes eines. Com que el vessant geomètric de les insercions ja havia quedat prou detallat, ara seguirem a partir dels punts suspensius del guió de ***GINSERT*** i ens centrarem amb l'assignació de valors provisionals als atributs, per passar tot seguit a l'execució de ***RATREDIT***:

Comando: ***GINSERT***
Indique nombre de bloque o [?]: ***P-F***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***G***
.....
Pare: ***Abraham***
Fill: ***01***
Comando: ***GINSERT***
Indique nombre de bloque o [?] <P-F>:
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***G***
.....
Pare: ***01***
Fill: ***02***
.....
.....
.....
Comando: ***GINSERT***
Indique nombre de bloque o [?] <P-F>:
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***G***
.....
Pare: ***37***
Fill: ***38***
Comando: ***GINSERT***
Indique nombre de bloque o [?] <P-F>:
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***G***
.....
Pare: ***38***
Fill: ***Josep***

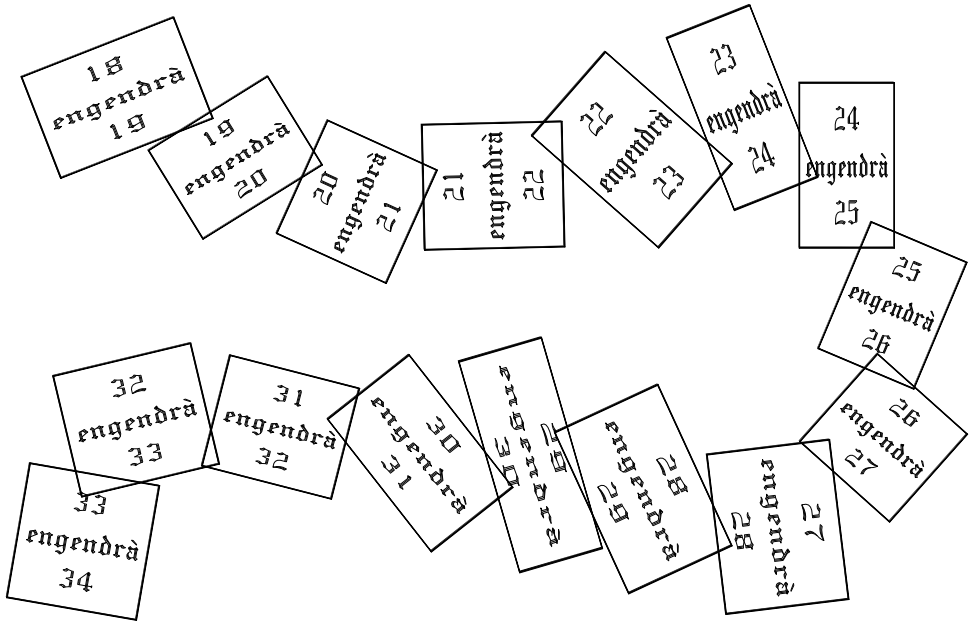


Figura 7.2

Comando: ***RATREDIT***
¿Editar atributos uno a uno? [Sí/No] <S>: ***N***
Realizando edición global de valores de atributos.
¿Editar sólo atributos visibles en pantalla? [Sí/No] <S>: ***N***
Indique especificación de nombre de bloque <>:*
Indique especificación de identificador de atributo <>:*
Indique especificación de valor de atributo <>:*
78 atributos designados.

Indique cadena a cambiar (INTRO para acabar): ***01***
Indique nueva cadena (INTRO para acabar): ***Isaac***
Indique cadena a cambiar (INTRO para acabar): ***02***
Indique nueva cadena (INTRO para acabar): ***Jacob***
.....
Indique cadena a cambiar (INTRO para acabar): ***26***
Indique nueva cadena (INTRO para acabar): ***Josias***

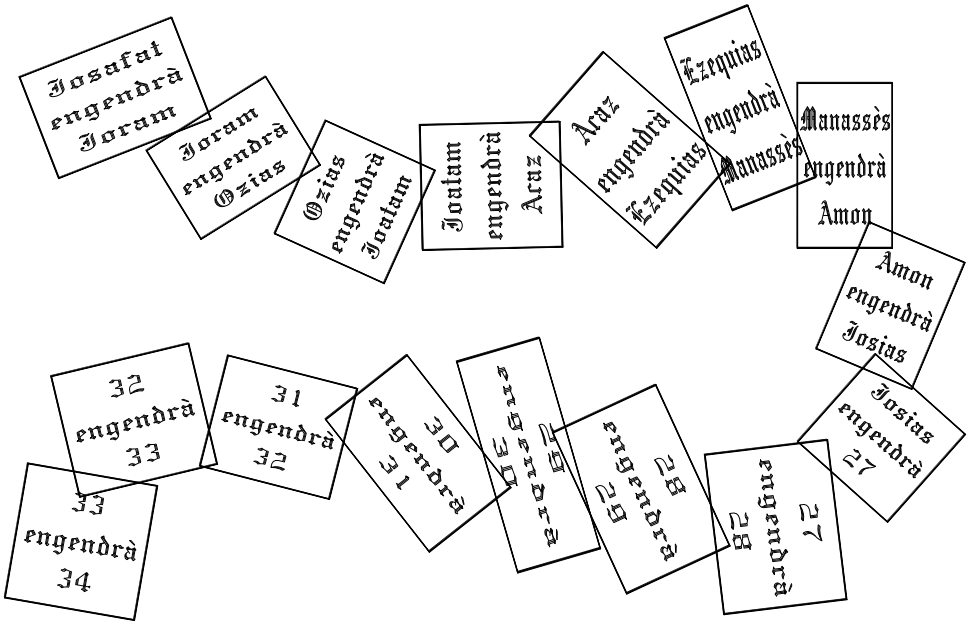


Figura 7.3

.....
Indique cadena a cambiar (INTRO para acabar): ***37***
Indique nueva cadena (INTRO para acabar): ***Mattan***
Indique cadena a cambiar (INTRO para acabar): ***38***
Indique nueva cadena (INTRO para acabar): ***Jacob***
Indique cadena a cambiar (INTRO para acabar):
Indique nueva cadena (INTRO para acabar):
Comando:

Passant ja a **C:GINSERT**, **C:RATREDIT** i altres funcions auxiliars, comencem aclarint que el propòsit inicial d'aquestes aplicacions era el d'il·lustrar, en el context d'una assignatura de lliure elecció, complementària d'ELEMENTS DE CAD, que havia de tocar la programació amb AutoLISP i que no va arribar a veure la llum per raons que no fan al cas, les possibilitats d'ampliació del repertori d'ordres AutoCAD, en la línia preconitzada al començament del capítol. Aprofitats aquests materials per al *totum revolutum* que teniu entre mans, hem eliminat els parèntesis didàctics orientats a l'aprenentatge d'AutoLISP, el coneixement del qual per part del lector donem per fet, i ens limitarem a la descripció del camí recorregut, fent marrades pels punts crítis, discutint alternatives i justificant les solucions adoptades.

La primera consideració a fer és que, si no hagués estat per les qüestions formals (que l'eco en pantalla del diàleg amb l'usuari fos el més aproximat possible al que es dóna treballant directament amb AutoCAD, sense mitjancers) o col·laterals (per exemple, com assegurar, quan decidim de neutralitzar les últimes operacions realitzades, que les ordres ***DESHACER*** o ***H*** tractaran ***GINSERT*** i ***RATREDIT*** com si de debò fossin ordres AutoCAD estàndard), la cosa hagués estat força més senzilla. O, dit a la recíproca, que la major dificultat (parlem tant d'esforç personal com de línies de codi) correspon a aquests aspectes secundaris, perquè si n'haguéssim prescindit del tot, les dues funcions principals haurien pogut quedar-se en això:

```
(defun C:GINSERT (/ BLOC R1 ANG RR TEV)
  (setq BLOC (getstring "Indique nombre de bloque o [?]: ")
        R1 (initget 1 "Escala X Y Z Girar PEscala PX PY PZ PGirar")
        R1 (getpoint (strcat "Precise punto de inserción o "
                             "[Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ")))
```

```
  (if (= R1 "Girar")
      (progn
        (setq ANG (* (/ (getangle "Precise ángulo de rotación: ") PI) 180)
              TEV (getvar "TEXTEVAL"))
        (repeat 25 (setq RR (cons '(if (> (getvar "CMDACTIVE") 0) PAUSE) RR)))
        (command "SCP" "N" "Z" ANG "TEXTEVAL" 1)
        (eval (append '(command "-INSERT" BLOC "G" 0) RR))
        (command "TEXTEVAL" TEV "SCP" "PR"))
      (command "-INSERT" BLOC R1)))

(defun C:RATREDIT (/ R1 R3 R4 R5 R7 R8)
  (setq R1 (getstring "¿Editar atributos uno a uno? [Sí/No] <S>: "))
  (if (wcmatch (strcase R1) "S*,")
      (command "-ATREDIT" "")
      (while (not (and R7 (= R7 R8)))
             (command "-ATREDIT" "N" "N"
                      (if R3 R3 (setq R3 (getstring T)))
                      (if R4 R4 (setq R4 (getstring)))
                      (if R5 R5 (setq R5 (getstring T)))
                      (setq R7 (getstring T))
                      (setq R8 (getstring T))))))
```

En el cas de **C:GINSERT** el codi és prou expressiu de l'estratègia de resolució i només caldrà aclarir un parell de qüestions "tècniques". Comencem reproduint les dues preguntes amb què arrenca l'ordre original ***-INSERT***, que són inamovibles, i actuarem en funció de la resposta donada a la segona: únicament si l'usuari escull l'opció ***Girar*** contestant aquest segon requeriment (no s'hi val si ho deixa per a més endavant), tindrà ocasió de tastar el nostre invent; qualsevol altra resposta el tornarà a posar en mans de l'Editor de Dibuix, per seguir executant l'ordre de manera convencional. L'angle **ANG** amb què vulgui girar la inserció serà utilitzat per adoptar un sistema de coordenades amb la mateixa orientació (ordre ***SCP***, girant l'angle **ANG** al voltant de l'eix ***Z***), sota el qual abordarem l'execució de ***-INSERT*** amb l'opció ***Girar*** per establir, d'entrada, una rotació nul·la (equivalent a un gir **ANG** en el sistema precedent): en aquestes condicions, si entrem el punt d'inserció i tot seguit fem ús del recurs ***Esquina***, la determinació gràfica simultània dels factors d'escala $\mathbf{E_x}$ i $\mathbf{E_y}$ es produirà acomplint les expectatives que s'havien vist defraudades en l'accés directe a l'ordre ***-INSERT***, segons que vèiem en començar el capítol. Inserit el bloc, sols restarà restablir el sistema precedent (***SCP PRevio***) per deixar les coses com estaven.

En principi tot és ben senzill, però no podem passar per alt els dos punts foscos que anunciàvem més amunt:

- Després de dir que girem un angle zero, no és possible de preveure quants ínputs més li caldran a l'usuari per completar l'execució de ***-INSERT***, i no només per l'eventual existència d'atributs (al cap i a la fi, podríem esbrinar quants n'hi ha consultant a la taula de símbols la composició del bloc), sinó perquè ***-INSERT*** és extraordinàriament flexible pel que fa a l'ordre d'aportació de les dades geomètriques, mentre no introduïm el punt d'inserció, i en funció de l'itinerari necessitarem més o menys intervencions (això sense comptar que, al marge de les opcions ***PE***, ***PX***, ***PY***, ***PZ*** i ***PG***, sempre podem tornar sobre una dada ja subministrada i modificar-ne el valor). Aquesta incertesa, que es tradueix en una inassumible indeterminació del nombre d'arguments de la funció **command**, l'haurem de resoldre fent un dimensionat per excés (en el codi precedent s'ha adoptat el valor 25) i muntant un dispositiu capaç de detectar en temps real si ***-INSERT*** encara està en marxa o ja s'ha executat, aportant l'argument **PAUSE** en el primer cas o un valor **nil** en el segon: així, mentre l'ordre demani més dades, **PAUSE** serà la porta que permetrà l'usuari anar-les-hi subministrant; quan l'execució s'hagi acabat, la resta d'arguments equivaldran a pulsacions de la tecla <Esc> i no causaran cap efecte, tret de canvis de línia si ***CMDECHO*** **= 1** (si en comptes de **nil** fossin textos nuls "", el primer provocaria la repetició de l'ordre que precedeix **C:INSERT**). El senyal que informa de manera permanent de si hi ha alguna ordre en marxa és la variable de sistema ***CMDACTIVE*** i, per això, els (25) últims arguments de la funció **command**, després de **"-INSERT"**, **BLOC**, **"G"** i **0**, repetiran l'expressió **(if (> (getvar "CMDACTIVE") 0) PAUSE)**. Passa, però, que per no haver d'escriure **N** vegades (25 o les que faci falta) aquesta expressió, hem construït

```
   (command "-INSERT" BLOC "G" 0 (if (> (getvar "CMDACTIVE") 0) PAUSE) ...
                                 ... (if (> (getvar "CMDACTIVE") 0) PAUSE))
```

com una llista sense avaluar, obtinguda de la fusió de dues llistes també sense avaluar,

'(command "-INSERT" BLOC "G" 0) i **'((if (> (getvar "CMDACTIVE") 0) PAUSE) ... ... (if (> (getvar "CMDACTIVE") 0) PAUSE))**

llista a la qual, un cop formada, aplicarem **eval** perquè reconegui que **command** és el nom d'una funció predefinida i n'avalui els arguments (la resta d'elements de la llista). I per a què tot aquest circumloqui?: doncs per poder construir la segona sublista, que anomenem **RR**, fent

(repeat N (setq RR (cons '(if (> (getvar "CMDACTIVE") 0) PAUSE) RR)))

Fixeu-vos que, si no hagués estat per l'apòstrof (notació simplificada de **quote**, funció que inhibeix l'avaluació de l'expressió col·locada com a argument seu), hauríem obtingut una llista formada per **N** valors **nil**, perquè tots els elements haurien estat avaluats abans que s'unissin ambdues sublistes i, com és lògic, abans que **eval** avalués **'command** i desencadenés l'execució de l'ordre ***-INSERT***, mentre que just allò que preteníem era una avaluació argument per argument, a cavall de l'execució de l'ordre.

- La variable de sistema ***TEXTEVAL*** controla la manera com AutoCAD ha d'interpretar la resposta a una sol·licitud de text, si n'entrem un amb la forma d'expressió AutoLISP. Potser seria més didàctic comentar la seva acció sobre l'ordre ***TEXTO***, però des d'ACAD 2000 això ja no és possible perquè ***TEXTEVAL*** no té cap influència sobre aquelles ordres que incorporen un text al dibuix en temps real, caràcter a caràcter, i la dualitat existent fins la versió 14 entre ***TEXTO*** (ordre que només traslladava el text al dibuix quan el donàvem per acabat amb <Intro>) i ***TEXTODIN*** (que el traslladava caràcter a caràcter, tot i que sovint només provisionalment) ha desaparegut: a partir de la versió 15 es mantenen ambdues ordres però es comporten de forma idèntica, a la manera de l'antiga ***TEXTODIN*** (paradoxalment, en executar-les des d'AutoLISP recuperen l'especificitat perduda, raó per la qual **(command "TEXTO" ... PAUSE)** és sensible a ***TEXTEVAL*** pero **(command "TEXTODIN" ...)** no). De tota manera, aquesta és una bona excusa per anar al gra, centrant-nos en l'ordre ***-INSERT***. Per exemple, si inserim un bloc proveït d'atributs normals i en resposta a un dels requeriments de valor escrivim **(setq TXT (+ 1 2 3))**, el text que apareixerà dibuixat en pantalla, amb la resta del bloc, i el valor de la variable **TXT** dependran de ***TEXTEVAL***: si ***TEXTEVAL* = 0** (valor per defecte) AutoCAD interpretarà literalment aquesta resposta, a la inserció veurem el text **(setq TXT (+ 1 2 3))** i, si fem **!TXT**, el resultat serà **nil**; si ***TEXTEVAL* = 1**, AutoCAD prendrà el valor que resulti d'avaluar el text com a expressió AutoLISP, en el bloc figurarà el text **6** i, si fem **!TXT**, obtindrem el valor numèric **6**. Tot això, quan l'ordre s'executa mitjançant **command** i usem **PAUSE** com a argument d'aquesta funció, encara és més sibil·lí: **PAUSE** és una constant AutoLISP, el valor de la qual és el caràcter contrabarra \ (representat com a **"\\"**), però la funció predefinida **command** està programada perquè, en trobar l'argument **"\\"** (tant si aquest valor és explícit com si està representat per **PAUSE** o per qualsevol altra expressió), s'interrompi l'avaluació i l'usuari pugui entrar en temps real el valor requerit per l'ordre (a partir d'ACAD 2000, també es poden usar funcions interactives de tipus **get...** o bé **ssget**, **entsel** i **nentsel**), jugant-hi aquest caràcter el mateix paper que en una opció de menú.
  Doncs bé, en trobar **command** l'argument **PAUSE** quan l'ordre en execució espera un text, pot reaccionar de dues maneres: si ***TEXTEVAL* = 0**, el prendrà com un text format exclusivament pel caràcter contrabarra (com si haguéssim posat **"\\"** en comptes de **PAUSE**, i en aquest sentit es podria afirmar que ***TEXTEVAL* = 0**, més que impedir l'avaluació de **PAUSE** com a expressió AutoLISP, només l'avalua "a mitges" (en detecta el valor però no desencadena l'acció associada); si ***TEXTEVAL* = 1**, **PAUSE** interromp l'avaluació (igual que ho faria l'argument explícit **"\\"**), tot esperant que l'usuari escrigui un text.
  Així doncs, a l'hora d'avaluar **(command "-INSERT" ...)** caldrà assegurar-se que ***TEXTEVAL* = 1**, almenys quan el bloc que manipulem contingui atributs. Però, poseu atenció! A diferència de quan cursàvem l'ordre sense mitjancers, si ***TEXTEVAL* = 1** i l'usuari escriu un text que sintàcticament pugui considerar-se una expressió AutoLISP, **command** només podrà interpretar-la literalment. N'hi haurà prou a fer **(command "-INSERT" ... PAUSE)** i a escriure **(setq TXT (+ 1 2 3))** quan l'avaluació s'interrompi perquè assignem un valor a l'atribut: a la inserció llegirem aquest mateix text i no pas el valor **6**, i si després fem **!TXT** el resultat serà **nil**.
  ¿Això vol dir que, si apliquem **C:GINSERT** a blocs portadors d'atributs i escrivim expressions AutoLISP quan ***-INSERT*** ens en demani els valors, els textos dibuixats només podran ser transcripció literal d'aquestes expressions, sense opció a transferir al dibuix el resultat de la seva avaluació? Efectivament, però no pas perquè aconseguir això sigui impossible sinó per la complicació que suposaria:

de la mateixa manera que, quan ***TEXTEVAL* = 0** l'hem de passar a **1** per tal que **PAUSE** sigui interpretat correctament, quan d'entrada ja fos ***TEXTEVAL* = 1** podríem detectar el moment en què apareix en pantalla el missatge *Indique valores de atributo*, preludi dels requeriments concrets d'assignació de valor (usant la variable de sistema ***LASTPROMPT*** com a sentinella, s'hauria d'acomplir la condició **(= (getvar "LASTPROMPT") "Indique valores de atributo")**) i, en comptes d'usar **PAUSE** per captar l'entrada de l'usuari, ho faríem amb **(setq ATR (getstring T))**; si també s'acomplís **(wcmatch ATR "!*,(*")**, deduiríem que aquesta entrada era una expressió AutoLISP i lliuraríem a la funció **command** els arguments alternatius **(eval (read (substr ATR 2)))** o **(eval (read ATR))**, depenent de si l'expressió començava amb **!** (seguia una variable) o amb **(** (era una funció). No és només que tot plegat resulti massa aparatós, sinó que en el cas de ***-INSERT*** autolimitar-nos juga a favor de l'opció més probable a la pràctica: pot tenir sentit aprofitar els atributs d'un bloc per guardar-hi expressions AutoLISP (per figurar en el diagrama lògic que esquematitza un programa AutoLISP, per exemple, o per avaluar més endavant les expressions guardades, amb un programa que llegeixi el dibuix), però no en té massa molestar-se a escriure una expressió AutoLISP amb la idea d'assignar a un atribut el seu valor, podent escriure aquest valor directament.

Aclarit això, passem a les dues qüestions formals que havíem esmentat. Començarem pel tractament que cal donar a **C:GINSERT** per tal que, quan decidim revocar-la fent ús de l'ordre ***H*** o, més genèricament, de l'ordre ***DESHACER*** amb un argument numèric **N** (en què **N** > <nombre d'ordres executades després de ***GINSERT***>), totes les accions a càrrec d'aquesta funció disfressada d'ordre AutoCAD quedin neutralitzades. De fet, sota el nom **C:GINSERT** (o el malnom ***GINSERT***, com vulgueu), s'aixopluguen 5 accions (***SCP***, ***TEXTEVAL***, ***-INSERT***, ***TEXTEVAL*** i ***SCP***, entre ordres i accessos a variables de sistema) si considerem el camí crític corresponent a **R1 = "Girar"**, i en la versió definitiva veureu que en són 8 (***TEXTEVAL***, ***CMDECHO***, ***SCP***, ***-INSERT***, ***CMDECHO***, ***CMDECHO***, ***TEXTEVAL*** i ***SCP***, i un ***CMDECHO*** final que pot quedar al marge), i caldrà encotillar-les entre ***DESHACER Inicio*** i ***DESHACER Fin***: tenir activat el mode ***DESHACER Auto*** sols garanteix que AutoCAD posarà "d'ofici" aquesta cotilla a qualsevol opció de menú que inclogui diverses accions, però en una aplicació AutoLISP cal intervenir-hi explícitament.

L'altra qüestió a tenir en compte és el control de l'eco del diàleg en la finestra de text: jugant hàbilment amb la variable de sistema ***CMDECHO*** hem de procurar que apareguin les intervencions d'ambdós interlocutors (els requeriments i avisos d'AutoCAD, d'una banda, i els ínputs escrits de l'usuari, de l'altra) però sense repeticions. I aquí és on comença el problema, perquè es produeix un conflicte d'interessos entre el primer **(setvar "CMDECHO" 0)** i el **(command "DESHACER" "I")** inicial: per silenciar aquesta última expressió caldria desactivar l'eco abans, però si inicialment teníem ***CMDECHO* = 1** i just després d'executar **C:GINSERT** decidim revocar-la amb ***H*** (o, **N** ordres després, decidim revocar-la amb ***DESHACER*** <**N+1**>), ens quedarem amb ***CMDECHO* = 0** i caldrà un **(setvar "CMDECHO" 1)** per deixar-ho tot igual. Aquesta consideració fa que ens decidim a desactivar l'eco després del ***DESHACER I***, però això provocarà que l'arrencada de l'ordre quedi així en la finestra de text:

Comando: ***GINSERT***
Indique nombre de bloque o [?]: ***<nom del bloc>***
Precise punto de inserción o [Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ***G***
Precise ángulo de rotación <0>: ***<clic sobre C>***
Designe segundo punto: ***<clic sobre D>***
DESHACER Indique el número de operaciones a deshacer o [Auto/Control/Inicio/Fin/Marca/Retorno] <1>: ***I***
Comando:

El problema no és l'eco *DESHACER* ni *Indique el número de operaciones a deshacer o [Auto/Control/Inicio/Fin/Marca/Retorno] <1>:* (de fet són inseparables, en el sentit que qualsevol acció posterior a l'argument **"DESHACER"**, encara que s'avalués abans del lliurament de **"I"**, no podria evitar l'aparició del submenú d'opcions de ***DESHACER***), que podem inhibir desactivant ***CMDECHO*** així

```
(defun BS (I) (repeat I (princ "\10 \10")))

(defun BLANC () (princ "\r") (repeat 100 (princ " ")) (princ "\r") (princ))

(defun C:GINSERT (/ ECO BLOC R1 ANG RR TEV)
  (setq ECO (getvar "CMDECHO")
        BLOC (getstring "Indique nombre de bloque o [?]: ")
```

```
      R1 (initget 1 "Escala X Y Z Girar PEscala PX PY PZ PGirar")
      R1 (getpoint (strcat "Precise punto de inserción o "
                           "[Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ")))
  (setvar "CMDECHO" 0)
  (if (= R1 "Girar")
      (progn
        (setq ANG (* (/ (getangle "Precise ángulo de rotación: ") PI) 180)
              TEV (getvar "TEXTEVAL"))
        (repeat 25 (setq RR (cons '(if (> (getvar "CMDACTIVE") 0)
                                       PAUSE
                                       (progn
                                         (if (= (getvar "CMDECHO") 1)
                                             (progn
                                               (setvar "CMDECHO" 0)
                                               (BLANC)))
                                         ()))
                                  RR)))
        (command "DESHACER"
                 (progn
                   (setvar "CMDECHO" 1)
                   "I"))
        (BLANC)
        (setvar "CMDECHO" 0)
        (command "SCP" "N" "Z" ANG "TEXTEVAL" 1)
        (eval (append '(command "-INSERT" BLOC "G" 0
                                (progn
                                  (setvar "CMDECHO" 1)
                                  (prompt "Precise punto de inserción: ")
                                  PAUSE))
                      RR))
        (command "TEXTEVAL" TEV "SCP" "PR" "DESHACER" "F"))
      (command (progn
                 (setvar "CMDECHO" 1)
                 "-INSERT")
               (progn
                 (BS 40)
                 (setvar "CMDECHO" 0)
                 BLOC)
               R1))
  (setvar "CMDECHO" ECO)
  (princ))
```

o reescriure'ls amb espais en blanc, mitjançant la funció **BS** (**BS** i **BLANC** són les mateixes i no les repetim), així

```
(defun C:GINSERT (/ ECO BLOC R1 ANG RR TEV)
  (setq ECO (getvar "CMDECHO")
        BLOC (getstring "Indique nombre de bloque o [?]: ")
        R1 (initget 1 "Escala X Y Z Girar PEscala PX PY PZ PGirar")
        R1 (getpoint (strcat "Precise punto de inserción o "
                             "[Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ")))
  (if (= R1 "Girar")
      (progn
        (setq ANG (* (/ (getangle "Precise ángulo de rotación: ") PI) 180)
              TEV (getvar "TEXTEVAL"))
        (repeat 25 (setq RR (cons '(if (> (getvar "CMDACTIVE") 0)
                                       PAUSE
                                       (progn
                                         (if (= (getvar "CMDECHO") 1)
                                             (progn
                                               (setvar "CMDECHO" 0)
                                               (BLANC)))
                                         ()))
                                  RR)))
        (command "DESHACER"
                 (progn
                   (if (= ECO 1) (BS 100))
                   "I"))
```

```
        (if (= ECO 1) (BLANC))
        (setvar "CMDECHO" 0)
        (command "SCP" "N" "Z" ANG "TEXTEVAL" 1)
        (eval (append '(command "-INSERT" BLOC "G" 0
                                (progn
                                  (setvar "CMDECHO" 1)
                                  (prompt "Precise punto de inserción: ")
                                  PAUSE))
                      RR))
        (command "TEXTEVAL" TEV "SCP" "PR" "DESHACER" "F"))
      (command "-INSERT"
               (progn
                 (if (= ECO 1)
                     (progn
                       (BS 40)
                       (setvar "CMDECHO" 0)))
                 BLOC)
               R1))
  (setvar "CMDECHO" ECO)
  (princ))
```

sinó l'eco de l'entrada ***I*** que, en produir-se seguit d'un canvi de línia, no hi ha possibilitat de neutralitzar per mitjans convencionals. Per salvar aquest escull, un podria caure en el parany d'agafar el rave per les fulles, partir de l'esquema

```
(defun C:GINSERT (/ ECO BLOC R1 ANG RR TEV)
  (setq ECO (getvar "CMDECHO")
        BLOC (getstring "Indique nombre de bloque o [?]: ")
        R1 (initget 1 "Escala X Y Z Girar PEscala PX PY PZ PGirar")
        R1 (getpoint (strcat "Precise punto de inserción o "
                             "[Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: ")))
  (setvar "CMDECHO" 0)
  (if (= R1 "Girar")
      (progn
        (setq ANG (* (/ (getangle "Precise ángulo de rotación: ") PI) 180)
              TEV (getvar "TEXTEVAL"))
        (repeat 25 (setq RR (cons '(if (> (getvar "CMDACTIVE") 0)
                                       PAUSE
                                       (progn
                                         (if (= (getvar "CMDECHO") 1)
                                             (progn
                                               (setvar "CMDECHO" 0)
                                               (BLANC)))
                                         ()))
                                  RR)))
        (command "DESHACER" "I" "SCP" "N" "Z" ANG "TEXTEVAL" 1)
        (eval (append '(command "-INSERT" BLOC "G" 0
                                (progn
                                  (setvar "CMDECHO" 1)
                                  (prompt "Precise punto de inserción: ")
                                  PAUSE))
                      RR))
        (command "TEXTEVAL" TEV "SCP" "PR" "DESHACER" "F"))
      (command "-INSERT" BLOC R1))
  (setvar "CMDECHO" ECO)
  (princ))
```

i passar del problema que una desactivació prèvia de ***CMDECHO*** pugués causar sobre l'acció de ***H*** o ***DESHACER***. O, més ben dit, de pensar a resoldre'l fora de **C:GINSERT**, mitjançant unes funcions *ad hoc*, **C:H** i **C:DESHACER**, que substituissin les ordres originals ***H*** i ***DESHACER*** (prèviament posades fora de servei per ***ANULADEF***) i fossin capaces de copsar si l'ordre o l'última de les ordres revocades (la primera, en el sentit de progressió en el dibuix) era ***GINSERT*** o qualsevol altra funció **C:** amb els mateixos síntomes. Naturalment, en les noves ***H*** i ***DESHACER*** també caldria tener cura de l'eco en la finestra de text, amb el consegüent perill d'entrar en algun cercle viciós. Doncs bé, a tall d'anècdota l'autor confessa haver arribat a embolicar tant la troca que, en el dispositiu adoptat d'una banda s'hi implicava la variable de sistema ***USERS1***, per anotar-hi les ordres o funcions **C:** que s'anessin executant

(en ser una variable de sistema, la seqüència s'actualitzava automàticament quan fèiem ***.H***, ***.DESHACER*** o ***REHACER***), de l'altra calia recórrer a Visual LISP per tal d'instal·lar dos reactius del tipus **vlr-COMMAND-reactor**, sensibles als successos **:vlr-CommandWillStart** i **:vlr-CommandEnded** (entre d'altres missions s'encarregaven d'efectuar l'anotació indiferenciada d'ordres bàsiques, també automàticament, ja que les funcions **C:** se'n cuidaven elles mateixes) i, com a exemple de manipulació rocambolesca, la nova ***DESHACER*** <**N**> amagava entre bastidors un procés en tres etapes, per tal d'esbrinar si l'última ordre a revocar (la primera, segons com es miri) era una ordre bàsica o una funció **C:** i si en el segon supòsit, just després d'executar-la, ***CMDECHO*** restava activada: de primer feia ***.DESHACER*** <**N-1**> i prenia nota de ***CMDECHO***, tot seguit feia ***REHACER*** i, finalment, ***.DESHACER*** <**N**>. Tot això, només per recompondre l'eco de les ordres revocades i per acabar decidint si calia una restauració **(setvar "CMDECHO" 1)**.

De vegades, complicar-se una mica la vida navegant amb un rumb equivocat permet de fer troballes interessants, sempre que donem el cop de timó a temps: aquí només calia adonar-se que, posats a usar aquests reactius, tenia més sentit aplicar-los directament a la neutralització d'uns ecos irreductibles per altres procediments (aprofitant que el succés **:vlr-CommandEnded** es produeix abans del canvi de línia), i no entestar-se a desactivar ***CMDECHO*** abans d'inaugurar oficialment la nova ordre amb ***DESHACER I***, evitant aquests ecos però a costa d'ampliar l'abast del problema petit i, a sobre, crear-ne uns altres de més entitat.

Però ja portem massa parlant de **C:GINSERT** (en la versió definitiva del codi, al final del capítol, trobareu la funció **-ECO-2**, tambe utilitzada des de **C:RATREDIT**) i potser és hora d'ocuparse també de l'altra funció.

En el cas de **C:RATREDIT** podeu veure que hem tirat pel dret: si la resposta a la primera pregunta és afirmativa, ens limitem a posar en marxa ***-ATREDIT*** i a cedir-ne el control a l'Editor de Dibuix (no res a objectar, perquè ja havíem quedat que no tenia sentit "motoritzar" aquesta via); però quan és negativa i anem a una edició global d'atributs, l'afany de simplicitat ens ha dut a pecar d'expeditius imposant el ***no*** com a única resposta a *¿Editar sólo atributos visibles en pantalla?*, perquè el ***sí*** potser no tindrà massa interès quan, com a l'exemple, el dibuix estigui constituït exclusivament per les insercions amb els atributs que volem editar, però sí en d'altres casos. Així doncs, la primera fase expansiva en la progressiva complicació del codi encara no respondrà als aspectes formals que comentàvem, sinó a la decisió d'incorporar a l'edició global repetitiva la selecció gràfica dels atributs editables. La dificultat que això comporta és doble:

- El nombre d'arguments de la funció **command** en la primera iteració a càrrec de **while** (aquella en què l'usuari hagi de seleccionar interactivament els atributs editables) no està predeterminat.
- En les iteracions que següeixen, no és cap qüestió trivial decidir com hauran de reaparèixer els atributs seleccionats en la primera, perquè ni ***-ATREDIT*** accepta molts dels arguments que funcionen satisfactòriament en altres ordres d'edició (no accepta els mateixos tipus de valor o no n'accepta el mateix nombre) ni tots els arguments acceptats ens retornen els atributs en les mateixes condicions en què ho farien altres ordres.

Per superar el primer obstacle sense haver de recórrer aquí també a l'artifici **(eval (append '(command ...) ...))** usat a **C:GINSERT**, hem començat realitzant la selecció sota la cobertura de la funció **ssget**. Aquest n'és el codi, que incorpora la funció auxiliar **SEL-ATRS**:

```
(defun SEL-ATRS (/ SS NE N E LE)
  (setq SS (list (cons 0 "INSERT") (cons 66 1) (cons 2 R3))
        SS (if (wcmatch (strcase R2) "S*,") (ssget SS) (ssget "X" SS))
        NE (sslength SS) N -1)
  (repeat NE
    (setq N (1+ N) E (ssname SS N))
    (while (progn
             (setq E (entnext E) LE (entget E))
             (= (cdr (assoc 0 LE)) "ATTRIB"))
           (if (and (wcmatch (cdr (assoc 2 LE)) R4)
                    (wcmatch (cdr (assoc 1 LE)) R5))
             (setq AA (ssadd E AA)))))
  AA)
```

```
(defun C:RATREDIT (/ AA R1 R2 R3 R4 R5 R6 R7 R8)
  (setq AA (ssadd)
        R1 (getstring "¿Editar atributos uno a uno? [Sí/No] <S>: "))
  (if (wcmatch (strcase R1) "S*,")
      (command "-ATREDIT" "")
      (while (not (and R7 (= R7 R8)))
             (command "-ATREDIT" "N"
                      (progn
                        (if R2 R2 (setq R2 (getstring)))
                        "")
                      (if R3 R3 (setq R3 (getstring T) R3 (if (= R3 "") "*" R3)))
                      (if R4 R4 (setq R4 (getstring) R4 (if (= R4 "") "*" R4)))
                      (if R5 R5 (setq R5 (getstring T) R5 (if (= R5 "") "*" R5)))
                      (if R6 R6 (setq R6 (SEL-ATRS)))
                      (setq R7 (getstring T))
                      (setq R8 (getstring T))))))
```

La funció **ssget** ens permet d'efectuar una primera preselecció per tal que, d'allò que seleccionem gràficament (edició un a un) o de la totalitat del dibuix (edició global) únicament en restin les insercions de blocs amb atributs i un nom que respongui al perfil **R3**. I, ja sobre el conjunt d'insercions que n'haurà resultat, seguim garbellant fins a quedar-nos només amb aquells atributs l'identificador i el valor dels quals encaixin en els perfils **R4** i **R5**, respectivament. Observeu que a l'opció d'edició global no li calia concórrer en unió de l'altra a **SEL-ATRS** (li hagués sobrat l'argument **R6** de **command**) i, només per poder integrar-les en una mateixa bateria d'arguments, hem fet que la resposta que se li transmet a ***-ATREDIT*** sigui sempre ***sí*** ("" provoca l'acceptació d'aquesta resposta per defecte).

La solució no és satisfactòria en l'aspecte formal ni tampoc pel que fa als resultats: quan inserim blocs amb més d'un atribut, d'una banda poden quedar atrapats més atributs que no volíem i, de l'altra, sols una part dels seleccionats (volguts i no volguts) romandrà a la nostra disposició per canviar-ne el valor. Veiem-ho amb més detall.

El procés que posa en marxa **ssget** és una selecció d'objectes AutoCAD, en general, i encara que **SEL-ATRS** inclou un procés en què entren insercions de bloc i en surt un tipus particular de component que són els atributs, l'acció de successius filtres és posterior a la intervenció de l'usuari seleccionant objectes a l'àrea gràfica, raó per la qual aquest es trobarà:

- Que les invitacions diuen *Designe objetos:* i no *Designe atributos:*.
- Que la confirmació gràfica d'allò que realment ha estat seleccionat també denota que són blocs (i d'altres objectes principals) i no atributs (que només en són una part): quan fem clic sobre un atribut, és tota la inserció que es redibuixa en línia discontínua; per contra, quan una finestra ***Ventana*** envolta un atribut però no del tot el bloc al qual pertany, la selecció no prospera.
- Que, quan una operació de selecció hagi capturat objectes que no siguin insercions de bloc amb atributs, apareix el missatge <***M***> *filtrados* després de <***N***> *encontrados*, conseqüència de la preselecció (la llista de filtres de **ssget**).
- Que, en cloure aquestes operacions, es llegeix <***N***> *encontrado(s)* en comptes del balanç <***N***> *atributos designados*, corresponent <***N***> a insercions de bloc amb atributs (diferència entre objectes capturats i filtrats) i no pas a atributs.

Però no és només que els missatges i la confirmació gràfica de la selecció revelin que no treballem directament amb ***-ATREDIT*** sinó que, si ens limitem a filtrar un conjunt que reuneix de manera indiscriminada tots els atributs de les insercions seleccionades, adeqüant-lo als perfils **R4** i **R5**, la tria no haurà pogut ser massa acurada: tret que, havent-ne capturat per excés, el resultat de filtrar la captura segons aquests perfils coincideixi amb la selecció gràfica que hauríem volgut fer, serà freqüent tenir polissons a bord.

Amb independència d'aquest problema, quan treballem amb blocs de més d'un atribut n'hi ha un altre de més greu: la reticència de ***-ATREDIT*** a admetre més d'un atribut per bloc, tret que la selecció es faci sense mitjancers. Així, quan constituïm un conjunt de selecció amb **ssget** i el manipulem per ajustar-lo als requeriments de l'ordre (que tot sigui atributs, encara que n'hi pugui haver de més), només un atribut de cada inserció (el primer que localitzi **entnext**) restarà disponible. No és pas que els altres s'hagin quedat fora: en el conjunt de selecció hi són tots, però quan ***-ATREDIT*** n'ha trobat un (hagi o no de canviar de valor) ignora la resta.

Així que, per garantir la disponibilitat de tots els atributs caldria multiplicar els arguments: per assegurar-nos la jugada, podríem usar **N** conjunts de selecció, cadascun amb un únic atribut, en comptes d'un únic conjunt de selecció amb **N** atributs; o encara més senzill, utilitzar directament els **N** noms interns d'aquests atributs. Però una altra singularitat de ***-ATREDIT*** invalida aquesta sortida o, si més no, ens obliga a complicar l'estructura iterativa de **C:RATREDIT:** a diferència d'altres ordres d'edició que, quan s'executen des de **command** admeten un nombre indeterminat d'arguments de tipus conjunt de selecció o de tipus nom d'objecte (seguits de l'argument "" per cloure la selecció), ***-ATREDIT*** sols n'admet un (sense cap ""); i si cada vegada que s'executa l'ordre només li subministrem un atribut (embolcallat en un conjunt de selecció o a cara descoberta), caldrà executar-la **N** vegades. En conseqüència, dintre del bucle **while** mostrat abans caldria muntar-ne un altre que fés passar els **N** atributs seleccionats per l'adreçador **R7 → R8**. Com podreu veure tot seguit, aquest el bucle interior no pren la forma **while**, **repeat** o **foreach**, sinó que se serveix de **ATR**, funció gestora de la nova variable **AA**: **AA** ja no és un conjunt de selecció amb **N** atributs sinó una llista de **N** elements, als quals **ATR** n'assegura l'accés. Aquests elements poden ser conjunts de selecció amb un únic atribut o, simplement, els noms interns d'aquests atributs, i segons que adoptem una solució o una altra, la penúltima línia de la funció **SEL-ATRS**, que

abans era **(setq AA (ssadd E AA)))))**
ara serà **(setq AA (cons (ssadd E) AA)))))**
o bé **(setq AA (cons E AA)))))**

No reproduïm la funció **SEL-ATRS** sencera perquè res no ha variat, tret d'aquesta línia, i ens limitarem a presentar **ATR** i la nova **C:RATREDIT:**

```
(defun ATR (M)
  (setq N (1+ M) R6 (nth N AA)))

(defun C:RATREDIT (/ AA R1 R2 R3 R4 R5 R6 R7 R8 N)
  (setq R1 (getstring "¿Editar atributos uno a uno? [Sí/No] <S>: "))
  (if (wcmatch (strcase R1) "S*,")
      (command "-ATREDIT" "")
      (while (not (and R7 (= R7 R8)))
             (command "-ATREDIT" "N"
                      (progn
                        (if R2 R2 (setq R2 (getstring)))
                        "")
                      (if R3 R3 (setq R3 (getstring T) R3 (if (= R3 "") "*" R3)))
                      (if R4 R4 (setq R4 (getstring) R4 (if (= R4 "") "*" R4)))
                      (if R5 R5 (setq R5 (getstring T) R5 (if (= R5 "") "*" R5)))
                      (if R7
                          (if (ATR N) R6 (ATR -1))
                          (if (SEL-ATRS) (ATR -1)))
                      (if (= N 0)
                          (setq R7 (getstring T))
                          R7)
                      (if (= N 0)
                          (setq R8 (getstring T))
                          R8)))))
```

Ja hem aconseguit que tots els atributs seleccionats a **SEL-ATRS** (que ja sabem que poden no coincidir del tot amb els que volíem seleccionar) siguin susceptibles d'edició, però el marc visual de l'execució ha empitjorat: ens referim a l'eco de la selecció que, de ser únic i del tipus *<**N**> encontrado(s)* en cada iteració **while**, ha passat a **N** ecos amb el text *1 encontrado(s)*. Fins ara hem prescindit de les qüestions formals i ho seguirem fent fins arribar a la versió definitiva del codi, però amb aquesta en farem una excepció: de fet, aquests missatges (que sols són admissibles una vegada, just després de la selecció gràfica) són els únics que no hem aconseguit eliminar (ni depenen de ***CMDECHO*** ni donen ocasió de ser reescrits en blanc), i l'única cosa que podem fer és, d'una banda, "amagar-los sota la catifa" (introduint dos salts de línia abans de la invitació *Indique cadena a cambiar,* de manera que no apareguin en pantalla gràfica mentre la finestra de text inferior no superi les tres línies) i, de l'altra, minimitzar-ne la incidència i el nombre, aspectes aquests últims de què ens ocupem tot seguit.

En primer lloc, caldrà fer-se enrera d'una decisió que en un primer moment havia semblat encertada, per simplificadora: traslladar a **SEL-ATRS** les diferències de

tractament derivades de la resposta **R2** a *¿Editar sólo atributos visibles en pantalla?*, que a efectes de **command** unificàvem afirmativament per regularitzar el nombre d'arguments (**(command "-ATREDIT" "N" "" ...)**). De seguida veurem que la complicació que ha representat diversificar els camins dintre de **command** ha estat mínima, i a canvi la incidència dels ecos no volguts queda limitada a la resposta ***sí*** (selecció gràfica). Pel que fa a reduir el nombre de missatges en cada iteració **while**, la funció **SS-LIST** agruparà els atributs en una mínima quantitat de conjunts de selecció, constituïts de manera que cadascun d'ells tingui el màxim d'atributs però sense que hi coexisteixin atributs diferents d'un mateix bloc: d'aquesta manera aconseguirem que tots els atributs restin disponibles i que alhora l'eco quedi reduït a un mínim de línies. La llista d'aquests conjunts l'anomenarem **SS-L**, i en la presentació d'ara hi tornarem a incloure **SEL-ATRS**, que ha experimentat canvis substancials:

```
(defun SS-LIST (ATR-L / SS ATR AA BB B)
  (while ATR-L
         (setq SS (ssadd) AA () BB ())
         (foreach ATR ATR-L
                  (setq B (cdr (assoc 330 (entget ATR))))
                  (if (member B BB)
                      (setq AA (cons ATR AA))
                      (progn
                        (ssadd ATR SS)
                        (setq BB (cons B BB)))))
         (setq ATR-L (reverse AA)
               SS-L (append SS-L (list SS)))))

(defun SEL-ATRS (/ SS NE N E LE AA)
  (setq SS (ssget (list (cons 0 "INSERT") (cons 66 1) (cons 2 R3)))
        NE (sslength SS) N -1)
  (repeat NE
    (setq N (1+ N) E (ssname SS N))
    (while (progn
             (setq E (entnext E) LE (entget E))
             (= (cdr (assoc 0 LE)) "ATTRIB"))
           (if (and (wcmatch (cdr (assoc 2 LE)) R4)
                    (wcmatch (cdr (assoc 1 LE)) R5))
               (setq AA (cons E AA)))))
  (SS-LIST AA))

(defun ATR (M)
  (setq N (1+ M) R6 (nth N SS-L)))

(defun C:RATREDIT (/ R1 R2 R3 R4 R5 R6 R7 R8 N SS-L)
  (setq R1 (getstring "¿Editar atributos uno a uno? [Sí/No] <S>: "))
  (if (wcmatch (strcase R1) "S*,")
      (command "-ATREDIT" "")
      (while (not (and R7 (= R7 R8)))
             (command "-ATREDIT" "N"
                      (if R2 R2 (setq R2 (getstring)))
                      (if R3 R3 (setq R3 (getstring T) R3 (if (= R3 "") "*" R3)))
                      (if R4 R4 (setq R4 (getstring) R4 (if (= R4 "") "*" R4)))
                      (if R5 R5 (setq R5 (getstring T) R5 (if (= R5 "") "*" R5)))
                      (if (wcmatch (strcase R2) "S*,")
                          (if R7
                              (if (ATR N) R6 (ATR -1))
                              (if (SEL-ATRS) (ATR -1)))
                          (setq R7 (getstring T)))
                      (if (wcmatch (strcase R2) "S*,")
                          (if (= N 0)
                              (setq R7 (getstring T))
                              R7)
                          (setq R8 (getstring T)))
                      (if (wcmatch (strcase R2) "S*,")
                          (if (= N 0)
                              (setq R8 (getstring T))
                              R8))))))
```

Ja només queda per resoldre, com a qüestió de contingut, assolir el control total de la selecció gràfica d'atributs. I un cop més, aquesta fita comportarà fer-se enrera d'una decisió precipitada: en aquest cas, del recurs a la funció **ssget** com alternativa a l'artifici **(eval (append '(command ...) ...))** utilitzat a **C:GINSERT** per poder executar ***-INSERT*** amb un nombre variable d'arguments. No cal justificar l'esmentat dispositiu, perquè ja ho hem fet en comentar aquesta primera ordre modificada; sols destacarem que aquí tampoc podem preveure el nombre ni els tipus d'ínputs que l'usuari acabarà cursant per seleccionar els atributs editables (punts en les designacions individuals, en els vèrtexs de finestres -rectangulars o poligonals- i de voravius, o textos per activar les opcions corresponents), però amb la diferència que aquests arguments oberts no se situen al final en l'execució de ***-ATREDIT***, que haurà d'acabar amb el primer parell de respostes **R7** i **R8** (valor a substituir i valor substitut). La detecció del moment en què s'haurà completat la selecció i haurem d'assignar a **R7** la resposta de l'usuari a la primera d'aquestes preguntes, pot anar a càrrec de la variable de sistema ***LASTPROMPT*** (no oblidem que amb la substitució de **ssget** per una sèrie d'arguments **PAUSE** haurem recuperat l'especificitat dels missatges de requeriment i de confirmació, que acabaran amb *<**N**> atributos designados*), però la qüestió realment espinosa i que cal afrontar d'entrada és com hauríem de guardar (o com podríem recuperar) els atributs seleccionats, per poder-los utilitzar més endavant sense intervenció de l'usuari. Considerem diverses possibilitats:

- La primera que se'ns acut és no fer res més d'especial en la primera iteració, tret de la selecció gràfica d'atributs, i recórrer en les posteriors al mode de designació ***Previo*** (fent **(ssget "P")**, per exemple). Però aquest camí no porta enlloc: ni cap selecció d'atributs efectuada dins de l'ordre ***-ATREDIT*** pot ser captada fent ***Previo*** en l'ordre d'edició següent, perquè resulta inaprehensible (encara que aquesta segona ordre sigui tan genèrica com ***GIRA***, si hi havia alguna selecció prèvia a ***-ATREDIT*** la recuperarà com si fos l'última, i si no, treurà el missatge *No hay ningún conjunto de selección previo*), ni tampoc ***-ATREDIT*** admet aquesta opció a l'hora de seleccionar atributs (**Designación no válida* Requiere un punto o Ventana/úLTimo/Captura/Marco/Borde/polígonOV/polígonOC*).

- Emmagatzemar tots els ínputs heterogenis (punts o textos, inclòs el "" final) subministrats per l'usuari en la primera iteració, i recuperar-los en les següents, reproduint fil per randa totes les operacions de selecció per tal d'obtenir el mateix botí. Tècnicament no tindria cap dificultat: anomenant **II** la llista d'ínputs, tot i que no serviria de res l'intent
  (progn (setq I PAUSE II (append II (list I))) I)
  perquè, amb independència de la variable de sistema ***TEXTEVAL***, només en treuríem una llista composta per caràcters contrabarra, n'hi hauria prou a fer
  (progn (initget "Ventana úLTimo Capturar Marco Borde polígonOV polígonOC")
  (setq I (getpoint) I (if I I "") II (append II (list I))) I)
  per disposar d'una llista d'arguments que en les iteracions posteriors repetiria mecànicament les operacions de selecció i darrera de la qual assignaríem valors a **R7** i **R8**; i en la primera iteració no caldria ni estar pendents de la variable ***LASTPROMPT***, perquè amb l'acompliment de **(= I "")** ja podríem substituir **getpoint** per **getstring**. Ara bé, repetir escrupolosament aquestes operacions garantiria la identitat dels resultats?. Disortadament no, perquè la longitud dels atributs aniria canviant en modificar-ne el valors: si un atribut s'hagués escurçat, un clic en la mateixa posició que abans l'encertava de ple podria ser que ara ni el fregués; i si, per contra, augmentés la longitud, la mateixa finestra de tipus ***Ventana*** que abans l'encerclava podria ser que ara ja no donés l'abast.

- No usar encara la primera iteració per canviar valors d'atribut amb caràcter definitiu, sinó dedicar-la sencera (i no només la selecció gràfica) a definir de forma precisa el conjunt d'atributs susceptibles d'edició. La jugada podria ser aquesta: introduir com a valor a canviar un text nul i com a nou valor un text força improbable (**(setq R7 "" R8 "W7&9Z")**, per exemple), cosa que provocaria que tots els atributs seleccionats incorporessin aquest text com a prefix; a partir d'aquí, podríem rastrejar el dibuix i, seguint un procés similar al de la funció **SEL-ATRS** ja superada (però sense filtrar el perfil **R5**, no fos cas que resultés incompatible amb els textos modificats), constituir un conjunt de selecció amb tots els atributs en què el valor comencés per aquest text (usaríem el criteri **(wcmatch (cdr (assoc 1 LE)) "W7&9Z*")**). El procediment pot arribar a ser feixuc (si en el dibuix hi ha molts més atributs) i tampoc és fiable al cent per cent (que el text triat com a prefix sigui estadísticament improbable no vol dir que sigui impossible), però ens dóna la pista per donar forma a un altre de millor.

El que farem serà partir d'aquesta última idea però evitant un rastreig que podria alentir el procés i que, en anar més enllà dels atributs seleccionats, comportaria un risc: que acabessin participant a la festa atributs que no hi eren convidats. El sofisticat sensor que ens permetrà d'inventariar els atributs seleccionats és el dispositiu més semblant a un detector automàtic amb què compta AutoLISP (Visual LISP, per ser exactes): un reactiu. En aquest cas, usarem un reactiu associat a la base de dades del dibuix (**vlr-ACDB-reactor**). I, ja posats a tirar la casa per la finestra i a haver de suportar la inquietant demora que la presència obligada de l'expressió **(vl-load-com)** suposa per a l'operació de càrrega de l'aplicació que projectem (amb la funció **load**, que haurem de reclamar des de l'Editor de Dibuix, o bé integrant el codi en els arxius de càrrega automàtica **ACAD.LSP** o **ACADDOC.LSP**), també usarem reactius, ara associats a les ordres d'AutoCAD (**vlr-COMMAND-reactor**), per acabar de polir alguns aspectes formals de l'execució (l'eco del diàleg entre l'usuari i AutoCAD, a la finestra de text), tant per donar suport a **C:RATREDIT** com a **C:GINSERT**.

Si el succés anomenat **:vlr-ObjectOpenedForModify**, al qual condicionarem la funció de resposta **SEL-EDIT**, respongués a allò que suggereix el nom (és a dir, si just es produís quan l'objecte resultés designat dintre d'una ordre que té la missió de modificar-lo i no pas de clonar-lo o eliminar-lo), el procediment hauria estat més senzill: en el transcurs de la selecció, el reactiu s'hauria activat repetidament (tantes vegades com atributs designats) i **SEL-EDIT** aniria formant una llista ***AA*** d'atributs editables que **SS-LIST** convertiria en llista **SS-L** de conjunts d'atributs compatibles a efectes d'edició, de manera que aquesta iteració ja seria operativa i podria efectuar la primera edició **R7 → R8** ordenada per l'usuari. Per desgràcia, el significat de "objecte obert per ser modificat" s'ha d'entendre en el sentit més restrictiu que, si l'objecte seleccionat no acaba experimentant un canvi, el succés no s'haurà arribat a produir i, per tant, el reactiu a qui s'havia associat no s'activarà provocant l'execució de **SEL-EDIT**: en definitiva, fins que l'objecte no estigui a punt de ser modificat no tindrà lloc **:vlr-ObjectOpenedForModify**, i en el cas present aquesta situació no s'esdevindrà fins no haver definit un nou valor **R8** (encara que el valor a canviar, **R7**, encaixés en algun dels atributs designats, podria ser que el nou valor provoqués error com quan **R7 = R8 =** "" i, al capdevall, no se n'arribés a modificar cap); en aquest sentit, l'única diferència entre els successos tipificats com a **:vlr-ObjectOpenedForModify** i **:vlr-ObjectModified** és que el primer precedeix el segon, però van sempre aparellats i sense que entre l'un i l'altre hi hagi ocasió d'inserir-hi cap acció externa.

Això obligarà a complicar-ho tot una mica, ajornant l'edició efectiva d'atributs fins a la segona iteració ***-ATREDIT*** i posteriors: dedicarem la primera a consolidar la selecció gràfica, fent que tots els atributs designats es vegin afectats per un canvi de valor, condició "sine qua non" perquè s'activi el reactiu **ACDB** i a la llista ***AA*** que hauran anat creant les respostes **SEL-EDIT** hi figurin tots els noms interns. Inspirant-nos en la tercera de les estratègies descartades, però sense el risc que algun dels atributs no seleccionats acabi a ***AA*** (ara no depenem d'un rastreig extès a tot el dibuix), farem el següent:

- Assegurar-nos que el text a substituir figuri en tots el atributs designats. Ho resoldrem adoptant el text nul (""), que ***-ATREDIT*** localitza al començament de tots els atributs (incloent-hi els de valor nul).
- Adoptar com a valor substitut qualsevol text no nul (hem triat **"Z"** com podria ser qualsevol altre).
- Un cop activat el reactiu i obtinguda la resposta (i les llistes ***AA*** i **SS-L**), podríem consagrar la segona iteració a aplicar la modificació inversa **"Z" → ""** (***-ATREDIT*** prendria només la primera **Z** dels atributs editables, és a dir, la que havíem afegit en la primera iteració), però resulta més senzill de neutralitzar aquesta ***-ATREDIT*** preliminar amb l'ordre ***H***.

Posant en solfa el procés que hem descrit, tenim el codi següent:

```
(vl-load-com)
(vlr-remove-all)
(vlr-ACDB-reactor () '((:vlr-ObjectOpenedForModify . SEL-EDIT)))

(defun SEL-EDIT (NOM-REACTIU L-VLAOBJ/LSPOBJ)
  (if CTRL-SEL
      (setq *AA* (cons (cadr L-VLAOBJ/LSPOBJ) *AA*))
      (if *AA* (setq *AA* ()))))
```

```
(defun SS-LIST (ATR-L / SS ATR AA BB B)
  (while ATR-L
         (setq SS (ssadd) AA () BB ())
         (foreach ATR ATR-L
                  (setq B (cdr (assoc 330 (entget ATR))))
                  (if (member B BB)
                      (setq AA (cons ATR AA))
                      (progn
                        (ssadd ATR SS)
                        (setq BB (cons B BB)))))
         (setq ATR-L (reverse AA)
               SS-L (append SS-L (list SS)))))

(defun INPUT ()
  (if (> (getvar "CMDACTIVE") 0)
      (if (wcmatch (getvar "LASTPROMPT") "* atributos designados.")
          (if R7
              "Z"
              (setq R7 ""))
          (progn
            (if (not CTRL-SEL) (setq CTRL-SEL T))
            PAUSE))
      (if CTRL-SEL
          (progn
            (setq N (length (SS-LIST *AA*))
                  CTRL-SEL ())
            "H"))))

(defun C:RATREDIT (/ R1 R2 R3 R4 R5 R6 R7 R8 RR N M CTRL-SEL SS-L)
  (setq R1 (getstring "¿Editar atributos uno a uno? [Sí/No] <S>: "))
  (if (wcmatch (strcase R1) "S*,")
      (command "-ATREDIT" "")
      (progn
        (setq R2 (getstring
                   "\n¿Editar sólo atributos visibles en pantalla? [Sí/No] <S>: ")
              R3 (getstring
                   "\nIndique especificación de nombre de bloque <*>: " T)
              R4 (getstring
                   "\nIndique especificación de identificador de atributo <*>: ")
              R5 (getstring
                   "\nIndique especificación de valor de atributo <*>: " T))
        (if (wcmatch (strcase R2) "S*,")
            (eval (append '(command "-ATREDIT" R1 R2 R3 R4 R5)
                          (repeat 25 (setq RR (cons (list 'INPUT) RR)))))
            (setq N 1 R7 T))
        (while (/= R7 R8)
               (if (wcmatch (strcase R2) "S*,") (setq M -1))
               (repeat N (command "-ATREDIT" R1 R2 R3 R4 R5
                                  (if M
                                      (setq M (1+ M)
                                            R6 (nth M SS-L))
                                      (setq R7 (getstring T)))
                                  (if M
                                      (if (= M 0)
                                          (setq R7 (getstring T))
                                          R7)
                                      (setq R8 (getstring T)))
                                  (if M
                                      (if (= M 0)
                                          (setq R8 (getstring T))
                                          R8))))))))
```

El codi definitiu de **C:RATREDIT**, que presentarem conjuntament amb el de **C:GINSERT**, amb funcions auxiliars específiques i comunes, és bàsicament aquest que hem vist: els additaments obeeixen al propòsit (fins ara reprimit, per no interferir en la línia dircursiva principal) de mantenir en l'usuari la il·lusió que l'entorn de treball és l'Editor de Dibuix AutoCAD, sense la mediació de l'avaluador AutoLISP.

En allò que és substancial per al bon funcionament de la maquinària, només cal introduir una correcció, però com que probablement ja n'estem fent un gra massa amb una descripció en què no estalviem al lector vacil·lacions ni ensopegades, en comptes de convertir-la en l'enèsima estació d'aquest peculiar via crucis, reblant les explicacions amb una nova versió del codi, farem com a **C:GINSERT**, limitant-nos a expressions AutoLISP aïllades o curts fragments de codi, i mirarem ja cap a la versió íntegra i definitiva, al final del capítol. Cal estar preparats per a una contingència que no només pot causar error i provocar la interrupció de **C:RATREDIT** sinó que, en produir-se quan la resposta **R2** a *¿Editar sólo atributos visibles en pantalla?* és ***sí*** (selecció gràfica prèvia), podria deixar sense processar atributs perfectament editables: si la resposta **R5** a *Indique especificación de valor de atributo <*>* no ha estat ***** (és a dir, hem introduït un perfil de valor d'atribut més o menys restrictiu) i les cadenes que substituïm coincideixen amb allò que és específic d'aquest perfil, pot passar que en successives iteracions el nombre d'atributs editables vagi disminuint, fins que ja no en quedi cap; quan s'arribi a aquesta situació, la següent iteració provocarà la interrupció del procés i rebrem el missatge **No válido*; error: Función cancelada*, sense que haguem tingut ocasió d'acabar **C:RATREDIT** protocolàriament.

Explicat així, podria semblar que l'incident no té gran transcendència: si ja no podíem modificar més atributs, almenys pel que fa als d'un determinat perfil **R5** (per editar-ne amb un perfil diferent cal tornar a executar **C:ATREDIT**), ja ens anava bé que el procés s'aturés automàticament, i només hagués calgut maquillar una mica aquest final tan traumàtic, neutralitzant el missatge (tot i que amb la funció ***error*** no desapareixerà la nota prèvia **No válido**, que no forma part del missatge estàndard d'error d'AutoLISP ni tampoc es pot inhibir amb ***CMDECHO***). Però el problema es presenta a causa de **SS-LIST**, per organitzar els atributs editables en la llista de conjunts de selecció **SS-L**, perquè es pot donar el cas que, després d'unes quantes iteracions, en algun dels conjunts de **SS-L** ja no quedin atributs ajustats al perfil **R5** però que encara en quedin en conjunts situats a continuació: en aquest cas, la sobtada interrupció del procés deixaria sense cap oportunitat atributs que encara són editables. Amb un senzill exemple ho veurem clar: si volem editar un per un el atributs ***Pare*** i ***Fill*** dels blocs ***P-F*** inserits en el dibuix, limitant però l'actuació als de perfil ****a****, i dels visibles en pantalla només en seleccionem els dos primers de la Figura 7.1 (suposem que, un cop enllestit el dibuix, volem jugar una mica), el primer conjunt de **SS-L** guardarà els noms interns dels atributs de valor ***Joram*** i ***Ozias***, i el segon guardarà els de valor ***Josafat*** i ***Joram***, ja que tots aquests valors encaixaven d'entrada en el perfil ****a**;** si la primera edició interactiva consisteix a fer la substitució **"a" → "A"**, els valors referenciats a **SS-L** s'hauran transformat en ***JorAm*** i ***OziAs*** d'una banda, i ***JosAfat*** i ***JorAm*** de l'altra; tot i que ***JosAfat*** és perfectament editable, si no féssim res per posar-hi remei **C:RATREDIT** ens deixaria plantats en aquest punt, perquè el primer conjunt portat a ***-ATREDIT*** ja no tindria cap atribut que encaixés en el perfil ****a****.

Així doncs, com que el problema és que els atributs no poden concórrer plegats a ***-ATREDIT*** sinó que ho han de fer organitzats en conjunts de selecció que desfilen l'un darrera de l'altre, haurem d'actuar en un doble front:

- Evitar que salti la xispa, eliminant la confrontació dels atributs seleccionats gràficament amb el perfil **R5**. Com ho farem? Ben senzill, eliminant el perfil (és a dir, adoptant el perfil *****), amb el benentès que en la llista ***AA*** (que **SS-LIST** reconverteix en **S-L**) ja hi és implícit, aquest perfil: en la primera iteració, que fa d'assaig general (edició prèvia **"" → "Z"**), ***AA*** surt precisament com a resultat d'aplicar els perfils **R3**, **R4** i **R5** a la selecció gràfica realitzada.
- Assumir el control que li hem arrabassat a ***-ATREDIT***, cosa que farem rastrejant la llista **AA** (copiada de ***AA***) abans de cada iteració, amb el dispositiu

```
(setq OK ())
(foreach A AA
        (if (not OK)
            (setq OK (wcmatch (cdr (assoc 1 (entget A))) R5))))
```

  i interrompent el procés quan, després del rastreig, sigui **OK = nil**.

I, ja posats a intervenir quan **(= R2 "Sí")**, ¿no seria millor fer-ho en qualsevol circumstància?: ja hem vist que, quan prescindim de selecció gràfica i extenem l'acció de ***-ATREDIT*** a tot el dibuix, la fi del procés es pot confiar a la pròpia reacció de l'ordre davant d'una situació anòmala, però si ho controlem nosaltres tot plegat quedarà més polit; l'única objeció seria que això obliga a generalitzar l'edició prèvia **"" → "Z"** i a activar el reactiu **ACDB** també quan **(= R2 "No")**, però de tota manera acabarem fent-ho així per més raons, com no trigarem a veure.

Quan ens ocupàvem de **C:GINSERT** ja hi ha hagut ocasió d'exposar la problemàtica i tractament de diverses qüestions formals: l'eco en pantalla del diàleg AutoCAD-usuari, la consideració d'aquestes funcions com a unitats operacionals, als efectes de poder neutralitzar la seva acció com si fossin ordres bàsiques, i la interrelació d'ambdós problemes (necessitat d'ajornar la desactivació de ***CMDECHO*** a l'enregistrament ***DESHACER Inicio***). En relació a **C:RATREDIT** i pel que fa al primer tema, i en concret als missatges post-selecció, hauríem d'afegir dues precisions:

- Quan contestem ***sí*** a *¿Editar sólo atributos* visibles *en pantalla? [Sí/No] <S>*, només ens interessa anar llegint els missatges *<**N**> encontrado(s)*, després de cada operació de designació, i el balanç final *<**M**> atributos designados*, una vegada, en la primera iteració. Després d'això, cap altre missatge que no sigui *Indique cadena a cambiar* o *Indique nueva cadena* (requeriments, per cert, que hem hagut de reproduir, inhibint els originals amb **(setvar "CMDECHO" 0)**, per posar-hi l'afegitó *(INTRO para acabar)*), i menys encara si els valors **N** dels missatges intrusos *<**N**> encontrado(s)* no tenen res a veure amb els valors de la selecció feta en temps real sinó amb el nombre d'atributs que la funció auxiliar **SS-LIST** ha pogut encabir en cada conjunt editable. Disortadament, ja havíem avançat que aquests missatges no depenen de ***CMDECHO*** ni, pel canvi de línia associat, poden ser reescrits amb espais en blanc, de manera que l'únic recurs al nostre abast ha estat separar-los visualment del primer dels requeriments esmentats, amb un canvi de línia addicional (per a més inri, ha calgut puntejar cada salt, per forçar el canvi de línia mentre el text es llegeix des de pantalla gràfica).

- Quan contestem ***no***, el problema és el contrari. En aques cas, el text informatiu *<**M**> atributos designados* no apareix, tot i que caldria que ho fes en la primera iteració, entre els requeriments *Indique especificación de valor de atributo <*>* i *Indique cadena a cambiar (INTRO para acabar)*. N'és la causa que, tret de la selecció gràfica d'atributs que correspon a la via del ***sí***, ens resultava més còmode mantenir desactivada ***CMDECHO*** per evitar el segon eco de les respostes escrites, de difícil manipulació, malgrat la feina de reproduir els requeriments inhibits. Igual que hem fet amb aquests requeriments, també podem reproduir el missatge díscol, però ens en falta una peça essencial: la quantitat **M** d'atributs seleccionats, que no és una dada emmagatzemada en cap variable de sistema ni que sigui fàcilment recuperable (recordeu que l'opció ***Previo*** no serveix per capturar atributs designats prèviament), de manera que si volem aquesta informació ens l'haurem de treballar. I, com que ja teníem muntat el reactiu **ACDB**, farem que també s'activi quan **(= R2 "No")**: per bé que, de la llista ***AA*** (que en la versió definitiva hem rebatejat com ***SEL-LIST***, amb vistes a una explotació més general de la funció **SEL-EDIT** on es va formant) d'entrada només calia saber el nombre **M** d'elements i no pas els elements mateixos, l'interès a detectar si els atributs editables s'han exhaurit, abans que ho faci ***-ATREDIT***, és un motiu més per donar aquest pas. De tota manera, no hi fa res que les nostres pretensions siguin més o menys modestes: haurem de pagar el mateix preu que quan **(= R2 "Sí")** i era vital disposar dels atributs seleccionats, completant la primera iteració amb la substitució **"" → "Z"** sense la qual el succés **:vlr-ObjectOpenedForModify** no s'arribaria a produir i el reactiu no s'activaria. Així que, ves per on, s'ha tancat el cicle: de primer havíem forçat la unificació estructural de les dues respostes a *¿Editar sólo atributos visibles en pantalla?*, encara que cada una rebés un tracte diferenciat dins de **SEL-ATRS**; després, únicament la resposta ***sí*** accedia a aquesta funció, però ambdues seguien compartint el motor **while** (tot i que aquesta última havia de multiplicar el nombre de revolucions amb un sistema de transmissions que li permetés d'extendre l'edició als contenidors d'atributs restants); tot seguit, en substituir **SEL-ATRS** per la funció **SEL-EDIT** associada a un reactiu, trencàvem la closca **while** i obligàvem la resposta ***sí*** a participar en solitari en una mena d'assaig general, només després del qual la consideràvem en condicions d'aplegar-se amb la resposta ***no*** i compartir l'oportunitat de celebrar tantes representacions com el públic demandés; finalment, encara que per raons diverses, també el ***no*** ha acabat participant de l'assaig general i així podria semblar que les dues respostes segueixen un mateix itinerari, quan la realitat és que en cadascuna d'aquestes etapes (assaig i representacions) el tracte que reben és força individualitzat.

Pel que fa a **-ECO-1** i **-ECO-2** (funcions de resposta al reactiu **COMMAND**, associades respectivament als successos **:vlr-CommandWillStart** i **:vlr-CommandEnded**), la primera és exclusiva de **C:RATREDIT** i la segona s'utilitza també des de **C:GINSERT**, ens permeten esborrar l'eco ***-ATREDIT*** de la invocació a aquesta ordre (només caldrà

fer-ho quan els atributs s'hagin d'editar un per un) i l'eco ***I*** de l'opció ***Inicio*** de ***DESHACER***, i s'ha de dir que, sense elles i encara que sembli mentida, els jocs malabars que hauríem de fer amb la variable de sistema ***CMDECHO*** per eludir o emmascarar aquests ecos serien força més complicats. Si ens haguéssim limitat als nostres problemes, el seu codi seria tan senzill com això

```
(defun -ECO-1 (N-REACTIU L-ORDRE) (if CTRL--ECO (BS 9)))
(defun -ECO-2 (N-REACTIU L-ORDRE) (if CTRL--ECO (BS 2)))
```

però les hem programat amb un caràcter més genèric, per poder-les aprofitar en altres aplicacions. El mateix hem fet amb **SEL-EDIT**, funció de resposta al reactiu **ACDB:** per si anés bé utilitzar-la en un altre context (amb objectes que no siguin atributs, que potser convé guardar en conjunts de selecció i no en llistes) s'ha substituït el nom de la variable global ***AA*** per ***SEL-EDIT*** i, en paral·lel, hem afegit ***SEL-SET*** perquè des de cada aplicació es pugui triar el format més adient.

Una altra presència estranya en la versió definitiva és la funció auxiliar **MODIF**, usurpant l'assignació de valor a **R7**. Si us hi fixeu, **MODIF** no només s'arroga aquesta missió sinó també l'assignació a **R8**, amb independència que el valor de la funció acabi sent **R7**. La raó de tot això, que ja s'havia avançat al final de la quarta pàgina d'aquest capítol, és detectar el cas en què **(= R7 R8 "")**, l'únic que no accepta ***-ATREDIT***, i reconvertir-lo a **(= R7 R8 "Z")** (podria ser qualsevol altre text no nul) per evitar que la fi de l'ordre es vegi enterbolida pel missatge **No válido**. De fet, per suspendre les iteracions **while** i acabar **C:RATREDIT** n'hi ha prou a contestar qualsevol cosa a *Indique cadena a cambiar* i donar la mateixa resposta a *Indique nueva cadena*, i si això no s'enuncia en aquests termes i ho deixem en *(INTRO para acabar)* és per no marejar l'usuari.

No us haurà passat per alt l'existència de línies de codi pràcticament duplicades, amb la còpia sostreta a l'acció de l'avaluador AutoLISP i reduïda a la categoria de comentari per intervenció de l'encapçalament ;;. Comparant en cada cas original i còpia, us haureu adonat que la segona usa la funció **getstring** (i, si no l'usa directament, remet a alguna altra funció que ho fa), funció predefinida que en el primer s'ha substituït per la funció d'usuari **GETSTR**, i és segur que n'haureu tret l'entrellat: el codi conté dues versions alternatives, una de les quals (se suposa que la millor) està operativa, mentre que l'altra (en les línies posades fora de servei) no serà avaluada i només s'adjunta perquè l'usuari en pugui disposar al seu criteri. Però a l'hora de comparar-les no és indiscutible ni de bon tros la superioritat de la primera sobre la segona, i cal introduir matisos:

- L'ús de l'expressió **(getstring T)** per a les entrades que s'assignen a **R5**, **R7** i **R8** no reprodueix amb total fidelitat les regles del joc de l'ordre ***-ATREDIT*** (que assumeix un espai en blanc inicial com a resposta nul·la). Però no només aquesta fotesa ens ha dut a buscar-li un relleu a **getstring**, sinó una peculiaritat de comportament que comparteix amb **PAUSE:** determinats usos del teclat (fer servir les tecles de desplaçament amunt i avall o esborrar els caràcters escrits fins a deixar el text momentàniament en no res) no interfereixen en el bon funcionament de ***RATREDIT*** però, si més endavant volem revocar la nova ordre, ens adonarem que han invalidat l'acció del dispositiu ***DESHACER Inicio ... DESHACER Fin***, perquè cal recórrer a més d'una ***H***. En el cas de les assignaciona a **R3** i **R5**, a més, la movilitat cap enrera queda afectada si hem deixat espais en blanc, encara que després els haguem esborrat.
- D'altra banda, tot i que la nova **GETSTR** subsana aquests tres problemes (a costa de complicar el codi, certament), presenta alguns desavantatges, com ara deixar fora de servei la tecla <Del> i les de desplaçament endavant i endarrera. A més, si piquem la tecla <Esc> en resposta a *Indique cadena a cambiar* o *Indique nueva cadena*, la cancel·lació s'associa a l'aparatós missatge *ERROR en la aplicación: Interrupción desde el teclado*, en comptes del més discret **Cancelado**. Però la conseqüència més negativa, d'entre totes les derivades de l'ús de la funció **grread**, és haver de prescindir de les tecles de desplaçament amunt i avall: d'una banda només permetien recuperar textos introduïts sota el control directe de l'Editor de Dibuix (i així, no era possible capturar un text **R8** anterior, per exemple, per tal de substituir diverses cadenes per aquest mateix valor sense necessitat d'escriure'l repetidament) i, de l'altre, els textos recuperats no ho feien substituint els actuals sinó encadenant-s'hi, raó per la qual hem preferit inutilitzar aquestes tecles fent que els textos recuperats accidentalment siguin ignorats (només quan es limitin a un caràcter burlaran el sistema de seguretat i caldrà esborrar-los manualment).

Tot i que encara hi ha una "tercera via": si volem preservar la movilitat completa del cursor mentre escrivim l'ínput, i valorem la possibilitat de recuperar textos

precedents (en les mateixes condicions que quan treballem directament amb l'ordre ***-ATREDIT***, una mica per sota de les que ofereix **getstring**) per sobre de l'afectació del dispositiu ***DESHACER Inicio ... DESHACER Fin***, podem usar **PAUSE** per a l'entrada dels textos **R7** i **R8**. En aquest cas cal adoptar les variants marcades amb ;; (les que utilitzen **getstring**) i amb ;;;: aquestes consten de les funcions auxiliars **INT-R7** i **INT-R8** (que substitueixen **MODIF**) i d'uns canvis dràstics a l'expressió **(command "-ATREDIT" R1 R2 R3 R4 R5 ...)** que resulten més senzills d'implementar substituint-la en bloc. Aquesta expressió se'ns ha complicat a causa d'una molt desconcertant (i molesta) singularitat de **PAUSE**: si aquesta constant no se situa en posició terminal o assimilable, i la pregunta a contestar admet una resposta nul·la o n'inclou una per defecte, no cojuntural sinó predeterminada per AutoCAD, només serà utilitzada si la resposta és explícita; quan ens limitem a picar la tecla <Intro> (o <espai>, si s'escau) **PAUSE** passarà a ocupar-se de la pregunta següent, l'eco de la qual (el missatge) es visualitzarà encara que ***CMDECHO* = 0**. Podríem dir que, en els casos esmentats, una resposta nul·la "no consumeix torn en l'ús de **PAUSE**", i en el cas de ***-ATREDIT*** aquesta circumstància es presentarà donant respostes nul·les, mitjançant **PAUSE**, a **R3**, **R4** o **R5** (per acceptar els valors per defecte "*****"), o a **R7** (per introduir un text buit), sent aquest últim cas l'arrel del nostre conflicte. Si donem resposta nul·la al **PAUSE** obert darrera del missatge *Indique cadena a cambiar (INTRO para acabar)* passaran dues coses:

- Encara que ***CMDECHO* = 0**, apareixerà el missatge d'AutoCAD *Indique nueva cadena*, sense possibilitat d'ampliar-lo o substituir-lo.
- Quan contestem a aquesta pregunta ho farem sobre el mateix **PAUSE** (l'avaluació s'haurà interromput en aquest punt) i tampoc no podrem fer res per impedir que ***-ATREDIT*** rebi una altra resposta nul·la si l'usuari torna a polsar <Intro> o <espai> però, en detectar-ho, farem que s'incompleixi la condició **(/= R7 R8)** i clourem **C:RATREDIT**. Tanmateix, no haurem pogut evitar l'aparició de **No válido**.

Conscients d'aquestes limitacions, el millor serà concentrar els dos avisos *(INTRO para acabar)* en un missatge únic, que haurà de precedir *Indique cadena a cambiar* i no convidar explícitament a polsar <Intro>. I per reduir l'impacte psicològic de la cridanera alarma **No válido**, allunyem-la del *Comando:* que seguirà **C:RATREDIT** (com ja havíem fet amb els missatges <***N***> *encontrado(s)*) i no li donem més voltes.

A diferència d'aquesta tercera versió i de la funció **C:GINSERT**, en què calia usar **PAUSE** per donar cabuda a ínputs que tant podien representar opcions com punts, factors d'escala, angles o valors d'atribut, en les altres dues els ínputs es cursen mitjançant **getstring** o **grread**, i **PAUSE** només s'empra per donar suport a les operacions de selecció gràfica dels atributs a editar, així que no caldrà que fem ***TEXTEVAL* = 1** per tal que **PAUSE** sigui interpretat de manera correcta i no com el caràcter contrabarra. Més aviat al contrari: si usem l'alternativa ;; (**getstring** en comptes de **GETSTR**), haurem d'assegurar-nos que sigui ***TEXTEVAL* = 0** per evitar que aparegui el missatge *Imposible volver a entrar en LISP* cada vegada que la cadena a canviar o la nova cadena tinguin la forma d'expressió AutoLISP; de tota manera, aquest missatge és tan inofensiu com **No válido** i només seria molest per la seva pròpia presència, atès que únicament ens informa que la cadena ha estat interpretada literalment, que és just el que volem (com ja havíem manifestat a propòsit de **C:GINSERT**, una versió de **C:RATREDIT** amb capacitat per reconèixer quan un text és una expressió AutoLISP i, si ***TEXTEVAL* = 1**, avaluar-lo i lliurar-ne el resultat, complicaria molt les coses sense cap contrapartida en l'ordre pràctic, raó per la qual tampoc ens l'hem plantejada).

Com que amb aquestes disquisicions potser estàvem incomplint el propòsit anunciat d'obviar parèntesis didàctics sobre AutoLISP, atès que ens adrecem a lectors prou bregats programant en aquest llenguatge, donem pas d'una vegada al codi definitiu:

```
(vl-load-com)
(vlr-remove-all)
(vlr-COMMAND-reactor () '((:vlr-CommandWillStart . -ECO-1)
                          (:vlr-CommandEnded . -ECO-2)))
(vlr-ACDB-reactor () '((:vlr-ObjectOpenedForModify . SEL-EDIT)))

(defun -ECO-1 (N-REACTIU L-ORDRE)
  (if CTRL--ECO (BS (1+ (strlen (getcname (strcat "_" (car L-ORDRE))))))))

(defun -ECO-2 (N-REACTIU L-ORDRE)
  (if CTRL--ECO (BS (1+ (strlen CTRL--ECO)))))
```

```
(defun SEL-EDIT (NOM-REACTIU L-VLAOBJ/LSPOBJ)
  (if CTRL-SEL
      (progn
        (setq *SEL-LIST* (cons (cadr L-VLAOBJ/LSPOBJ) *SEL-LIST*)
              *SEL-SET* (if *SEL-SET* *SEL-SET* (SSADD)))
        (SSADD (cadr L-VLAOBJ/LSPOBJ) *SEL-SET*))
      (if *SEL-LIST* (setq *SEL-LIST* () *SEL-SET* ()))))

(defun ERR-LOCAL (MS)
  (setq *error* **ERROR**)
  (if (and R (wcmatch MS "*Función cancelada*")) (command))
  (setvar "CMDECHO" 0)
  (command "DESHACER" "F")
  (if TEV (setvar "TEXTEVAL" TEV))
  (setvar "CMDECHO" ECO)
  (princ))

(defun BS (I) (repeat I (princ "\10 \10")))

(defun BLANC ()
  (princ "\r") (repeat 100 (princ " ")) (princ "\r") (princ))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun NOM-BLOC ()
  (setq ANT-BLOC (getvar "INSNAME")
        ANT-BLOC (if (> ANT-BLOC "") ANT-BLOC)
        BLOC (getstring (strcat "Indique nombre de bloque o [?]"
                                (if ANT-BLOC
                                    (strcat " <" ANT-BLOC ">: ")
                                    ": ")) T)
        BLOC (if (= BLOC "?")
                 (command "INSERT"
                          (progn
                            (if (= (getvar "CMDECHO") 1)
                                (progn
                                  (BS 46)
                                  (setvar "CMDECHO" 0)))
                            "?")
                          (progn
                            (setvar "CMDECHO" 1)
                            (prompt "\nIndique bloque(s) a listar <*>: ")
                            PAUSE))
                 (if (= BLOC "")
                     (if ANT-BLOC
                         ANT-BLOC
                         (prompt "\nNombre de bloque no válido."))
                     (if (or (tblsearch "BLOCK" BLOC)
                             (findfile (strcat BLOC ".dwg")))
                         BLOC
                         (alert (RUTES)))))))

(defun INPUT-1 ()
  (graphscr)
  (initget 1 "Escala X Y Z Girar PEscala PX PY PZ PGirar")
  (setq R1 (getpoint (strcat "Precise punto de inserción o "
                             "[Escala/X/Y/Z/Girar/PEscala/PX/PY/PZ/PGirar]: "))))
```

```
(defun INPUT-2 ()
  (setq ANG (getangle "Precise ángulo de rotación: ")
        ANG (if (= (getvar "ANGDIR") 0)
                ANG
                (- ANG))
        ANG (if (= (getvar "AUNITS") 3)
                ANG
                (* (/ ANG PI) (if (= (getvar "AUNITS") 2) 200 180)))))

(defun INPUT-INSERT ()
  (if (> (getvar "CMDACTIVE") 0)
      PAUSE
      (progn
        (if (= (getvar "CMDECHO") 1)
            (progn
              (setvar "CMDECHO" 0)
              (BLANC)))
        ())))

(defun C:GINSERT (/ N ECO TEV CTRL--ECO ANT-BLOC BLOC R1 ANG R RR)
  (setq **ERROR** *error* *error* ERR-LOCAL
        N 25 ; Ajusta'l al nombre màxim d'ínputs previsible (atributs inclosos) +2
        ECO (getvar "CMDECHO"))
  (if (NOM-BLOC)
      (if (= (INPUT-1) "Girar")
          (progn
            (INPUT-2)
            (command "DESHACER"
                     (progn (if (= ECO 1)
                                (progn
                                  (BS 100)
                                  (setq CTRL--ECO "I")))
                            "I"))
            (if (= ECO 1)
                (progn
                  (BLANC)
                  (setq CTRL--ECO ())))
            (setq TEV (getvar "TEXTEVAL"))
            (setvar "TEXTEVAL" 1)
            (setvar "CMDECHO" 0)
            (command "SCP" "N" "Z" ANG)
            (eval (append '(command "INSERT" BLOC "G" 0
                                    (progn (setvar "CMDECHO" 1)
                                           (prompt "Precise punto de inserción: ")
                                           PAUSE))
                          (repeat N (setq RR (cons (list 'INPUT-INSERT) RR)))))
            (setvar "TEXTEVAL" TEV)
            (command "SCP" "PR" "DESHACER" "F"))
          (command "INSERT"
                   (progn (if (= ECO 1)
                              (progn
                                (BS (if ANT-BLOC (+ 43 (strlen ANT-BLOC)) 40))
                                (setvar "CMDECHO" 0)))
                          BLOC)
                   R1)))
  (setvar "CMDECHO" ECO)
  (setq *error* **ERROR**)
  (princ))

(defun SINO (MS / S/N)
  (while (not S/N)
         (setq S/N (getstring (strcat MS " [Sí/No] <S>: "))
               S/N (if (wcmatch (strcase S/N) ", ,S,SI,SÍ")
                       "Sí"
                       (if (wcmatch (strcase S/N) "N,NO")
                           "No"
                           (prompt "\nSí o No.\n"))))))
```

```
(defun TXT-1 ()
  (substr TXT 1 (1- (strlen TXT))))

(defun GETSTR (MS CR / TXT GR N T1 T2 REC)
  (prompt MS)
  (setq TXT "")
  (while (not (or (= (setq GR (grread)
                           N (if (= (car GR) 2)
                                 (progn
                                   (setq T1 T2 T2 (getvar "DATE"))
                                   (if (and T1 (< (- T2 T1) T-MAX))
                                       (if (not REC); Si el text recuperat és d'un
                                           (progn   ; caràcter serà indetectable i
                                             (BS 1) ; l'haurà d'esborrar l'usuari.
                                             (not (setq TXT (TXT-1) REC T))))
                                       (progn
                                         (setq REC ())
                                         (cadr GR))))
                                 (progn
                                   (setq REC ())
                                   (if (equal GR '(11 0))
                                       13))))
                     13)
                  (and (= N 32)
                       (< CR 2)
                       (or (= CR 0) (= TXT "")))))
        (if N
            (if (= N 8)
                (if (/= TXT "")
                    (progn
                      (BS 1)
                      (setq TXT (TXT-1))))
                (setq TXT (strcat TXT (princ (chr N)))))))
  (princ (if (= N 13) "\n" " "))
  TXT)

(defun RESP (MS CR)
  (GETSTR (strcat "\nIndique especificación de " MS " <*>: ") CR))
;;  (getstring (strcat "\nIndique especificación de " MS " <*>: ") CR))

(defun MODIF ()
  (prompt (if (= R2 "Sí")
              "·\n·"
              (if R8
                  ""
                  (progn
                    (textscr)
                    (strcat "\n" M " atributos designados.")))))
  (setq R T
        R7 (GETSTR "\nIndique cadena a cambiar (INTRO para acabar): " 1)
        R8 (GETSTR "Indique nueva cadena (INTRO para acabar): " 1))
;;  (setq R7 (getstring "\nIndique cadena a cambiar (INTRO para acabar): " T)
;;        R8 (getstring "Indique nueva cadena (INTRO para acabar): " T))
  (if (and (= R7 "") (= R8 ""))
      (setq R7 "Z" R8 R7)
      R7))

;;;(defun INT-R7 ()
;;;  (setq R T)
;;;  (prompt (if (= R2 "Sí")
;;;              "\n·"
;;;              (if R8
;;;                  ""
;;;                  (progn
;;;                    (textscr)
;;;                    (strcat "\n" M " atributos designados.")))))
```

```
;;;  (prompt (strcat "\nPara acabar, responda lo mismo a las dos preguntas >>"
;;;                  "\nIndique cadena a cambiar: "))
;;;  PAUSE)

;;;(defun INT-R8 (/ LP)
;;;  (setq LP (getvar "LASTPROMPT"))
;;;  (if (> (getvar "CMDACTIVE") 0)
;;;      (progn
;;;        (setq R7 (substr LP 27) +R8 T)
;;;        (prompt "Indique nueva cadena: ")
;;;        PAUSE)
;;;      (not (setq R7 "" R8 (substr LP (if (wcmatch LP "Indique nueva*") 23 50))
;;;                 R8 (if (wcmatch R8 " *") "" R8)))))

(defun SS-LIST (ATR-L / SS ATR AA BB B)
  (while ATR-L
         (setq SS (ssadd)
               AA () BB ())
         (foreach ATR ATR-L
                  (setq B (cdr (assoc 330 (entget ATR))))
                  (if (member B BB)
                      (setq AA (cons ATR AA))
                      (progn
                        (ssadd ATR SS)
                        (setq BB (cons B BB)))))
         (setq ATR-L (reverse AA)
               SS-L (append SS-L (list SS)))))

(defun INPUT-ATREDIT ()
  (if (> (getvar "CMDACTIVE") 0)
      (if (wcmatch (getvar "LASTPROMPT") "* atributos designados.")
          (if R7
              "Z"
              (progn
                (setvar "CMDECHO" 0)
                (BLANC)
                (setq R7 "")))
          (progn
            (if (not CTRL-SEL)
                (progn
                  (setq CTRL-SEL T)
                  (setvar "CMDECHO" 1)
                  (prompt "\nDesigne atributos: ")))
            PAUSE))
      (if CTRL-SEL
          (progn
            (setq AA *SEL-LIST*
                  N (length (SS-LIST AA))
                  CTRL-SEL ())
            "H"))))

(defun C:RATREDIT (/ N T-MAX ECO TEV CTRL--ECO CTRL-SEL R1 R2 R3 R4 R5 R6 R7 R8 R
                     RR M SS-L AA A OK) ; Amb l'alternativa ";;;" cal afegir +R8.
  (setq **ERROR** *error* *error* ERR-LOCAL
        N 25        ; Ajusta'l al nombre màxim d'ínputs previsible en la selecció.
        T-MAX 1.0E-6  ; Ajusta'l a un lapse de temps (getvar "DATE") comprés entre
        ECO (getvar "CMDECHO")     ; el que triga a aparèixer cada nou caràcter en
        R1 (SINO "\n¿Editar atributos uno a uno?")) ; mantenir polsada una tecla o
  (if (= R1 "Sí")     ; en usar les tecles amunt o avall per capturar les entrades
      (command (progn ; anteriors, en el bucle (while ... (grread) ...) de GETSTR.
                 (setq CTRL--ECO T)
                 "ATREDIT")
               (progn
                 (BLANC)
                 (setvar "CMDECHO" 0)
                 (setq CTRL--ECO ())
                 ""))
```

```
      (progn
        (prompt "\nRealizando edición global de valores de atributos.")
        (command "DESHACER"
                 (progn
                   (BS 100)
                   (setq CTRL--ECO "I")))
        (BLANC)
        (setvar "CMDECHO" 0)
        (setq TEV (getvar "TEXTEVAL") ; Només cal en l'alternativa ";;".
              CTRL--ECO ()
              R2 (SINO "\n¿Editar sólo atributos visibles en pantalla?")
              R3 (progn (if (= R2 "No") (textscr)) (RESP "nombre de bloque" 2))
              R4 (RESP "identificador de atributo" 0)
              R5 (RESP "valor de atributo" 1))
;;            R3 (progn (if (= R2 "No") (textscr)) (RESP "nombre de bloque" T))
;;            R4 (RESP "identificador de atributo" ())
;;            R5 (RESP "valor de atributo" T))
        (eval (append '(command  "ATREDIT" R1 R2 R3 R4
;;        (eval (append '(command  "TEXTEVAL" 0 "ATREDIT" R1 R2 R3 R4
                                  (progn
                                    (if (= R2 "No")
                                        (princ "0 atributos designados."))
                                    R5))
                      (if (= R2 "Sí")
                          (repeat N (setq RR (cons (list 'INPUT-ATREDIT) RR)))
                          '((progn (BS 23) "")
                            (progn (setq N 1 R7 T
                                         CTRL-SEL T) "Z")
                            (progn (setq AA *SEL-LIST*
                                         M (itoa (length AA))
                                         CTRL-SEL ()) "H")))))
        (while (/= R7 R8)
               (if (or (wcmatch R5 "`*,")
                       (not R)
                       (progn
                         (setq OK ())
                         (foreach A AA
                           (if (not OK)
                               (setq OK (wcmatch (cdr (assoc 1 (entget A))) R5))))
                         OK))
                   (if (= R2 "Sí") (setq M -1))
                   (progn
                     (setq N 0 R8 R7)
                     (if (= R2 "Sí") (princ "."))
                     (princ (strcat "\nTras el último reemplazo ya no quedan"
                                    " más atributos del perfil especificado."))))
               (repeat N (if (/= R7 R8) ; Només per evitar N-1 missatges del tipus
                             ; "<núm> encontrado(s)" abans d'acabar, si R2 = "Sí".
                             (command "ATREDIT" R1 R2 R3 R4
                                      (if (= R2 "Sí") "" R5)
                                      (if (= R2 "Sí")
                                          (setq M (1+ M)
                                                R6 (nth M SS-L))
                                          (MODIF))
                                      (if (= R2 "Sí")
                                          (if (= M 0) (MODIF) R7)
                                          R8)
                                      (if (= R2 "Sí") R8)))))
;;;                           (command "ATREDIT" R1 R2 R3 R4
;;;                                    (if (= R2 "Sí") "" R5)
;;;                                    (if (= R2 "Sí")
;;;                                        (setq M (1+ M)
;;;                                              R6 (nth M SS-L))
;;;                                        (INT-R7))
;;;                                    (if (= R2 "Sí")
;;;                                        (if (= M 0) (INT-R7) R7)
;;;                                        (INT-R8))
```

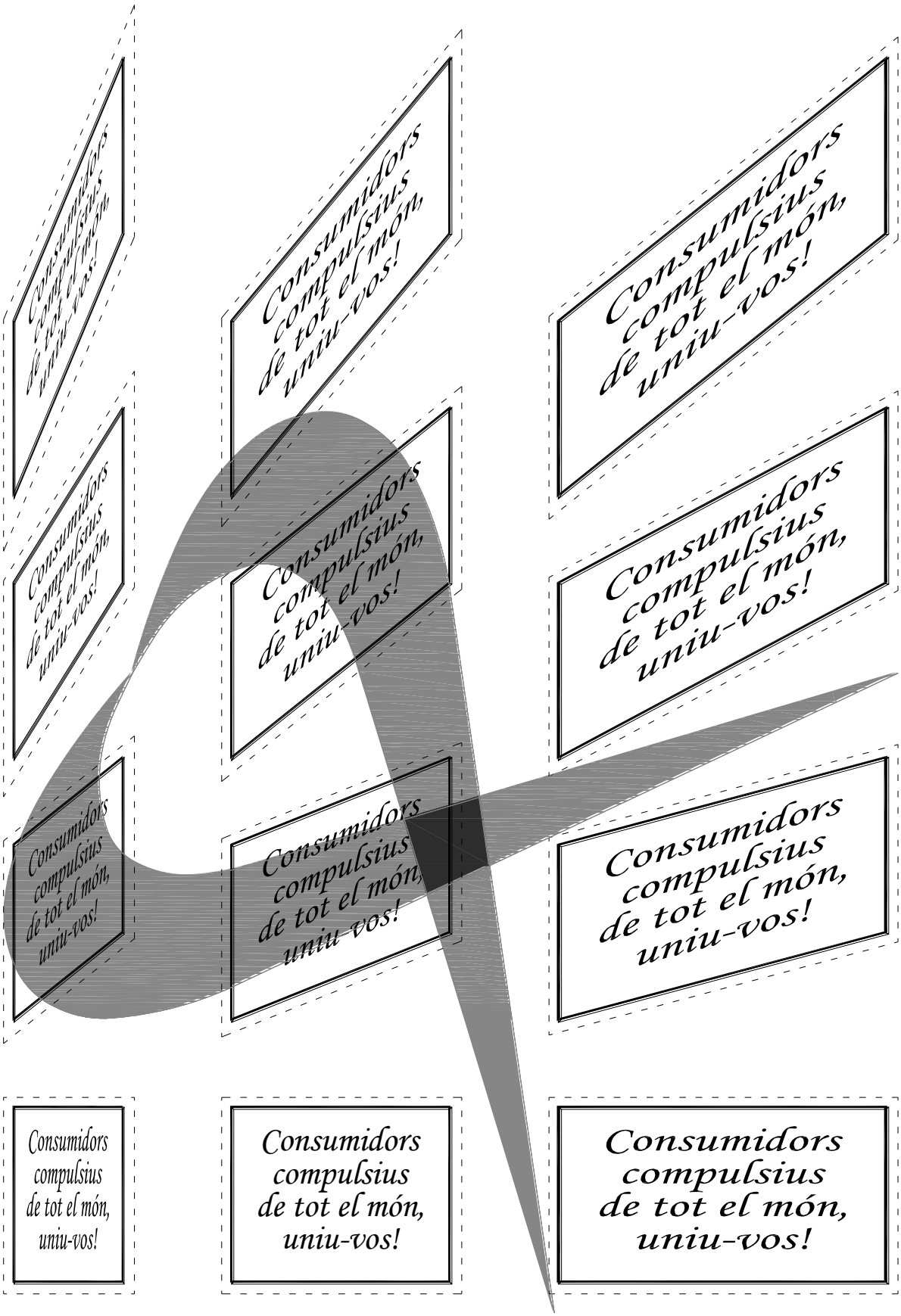
```
;;;                                          (if (= R2 "Sí")
;;;                                              (if (= M 0) (INT-R8) R8))
;;;                                          (if (or (= R2 "No") (= M 0))
;;;                                              (if +R8
;;;                                                  (setq R8 (substr
;;;                                                             (getvar "LASTPROMPT")
;;;                                                             23)
;;;                                                       R8 (if (= R8 " ") "" R8)
;;;                                                       +R8 ())
;;;                                              (prompt "·\n·")))))))
;;        (setvar "TEXTEVAL" TEV) ; Amb l'alternativa ";;", millor afegir això que
        (command "DESHACER" "F"))); fer (command "TEXTEVAL" TEV "DESHACER" "F"))).
  (setvar "CMDECHO" ECO)
  (setq *error* **ERROR**)
  (princ))
```

Per acabar, dos aclariments relatius a regles sintàctiques del llenguatge AutoLISP que són flexibles i que no s'han aplicat de manera uniforme en el codi:

- Sempre que l'execució d'una ordre admeti la dualitat d'expressar-se en l'àrea de text o mitjançant una finestra de diàleg, tant si això depèn del nom utilitzat (seria el cas de ***-BLOQUE*** i ***BLOQUE***, o de ***-INSERT*** i ***INSERT***) com si té a veure amb variables de sistema com ***FILEDIA*** (***GUARDAR***), ***ATTDIA*** (altre cop ***-INSERT*** i ***INSERT***, quan s'apliquen a blocs proveïts d'atributs) o ***CMDDIA*** (***SALTRAZ***)*, si la cridem des de **command** l'execució respondrà a la primera modalitat, amb independència del nom de l'ordre o del valor de la variable. ***-ATREDIT*** i ***ATREDIT*** també obeeixen aquesta norma: malgrat que la segona no té el mateix abast que la primera (a més d'obrir una finestra de diàleg), usada des de **command** s'hi identifica del tot. Així doncs, no ens ha d'estranyar que a les versions prèvies del codi hi figurin ***-INSERT*** i ***-ATREDIT*** però que a la definitiva haguem passat a ***INSERT*** i ***ATREDIT***: com a arguments de la funció **command**, ***INSERT*** i ***ATREDIT*** es comporten igual que ***-INSERT*** i ***-ATREDIT***.
- Tampoc ha d'estranyar que el missatge de sol·licitud de la funció **getstring** vagi en primer lloc i l'argument lògic **T** (admissió d'espais en blanc) en segon, quan segons els manuals d'AutoLISP l'ordre dels argument és just el contrari: com a d'altres funcions **get...** (com per exemple **getpoint**, amb els arguments opcionals punt base i missatge), si el primer argument és de tipus text AutoLISP assumirà que és el missatge.

---

* Pel que fa a ***ATTDIA*** i a ***CMDDIA***, això és aplicable a la versió 15 de referència (ACAD 2000) però no a les posteriors, en què: si ***ATTDIA*** = **1** l'assignació de valor als atributs es fa a través d'una finestra de diàleg, fins i tot quan ***-INSERT*** o ***INSERT*** s'executin des d'AutoLISP (a diferència del cas ***ATTDIA*** = **0**, aquesta operació no requerirà de cap argument, situació que es tradueix en ***CMDACTIVE*** = **0** i així serà captada per la nostra funció **INPUT-INSERT**, raó per la qual el codi manté plena vigència en la versió 16 -ACAD 2004-); substituïda l'ordre ***SALTRAZ*** per ***-TRAZADOR*** i ***TRAZADOR***, la variable ***CMDDIA*** resta obsoleta. Aprofitem la menció d'ACAD 2004 per assenyalar que la versió 16 ha tancat el parèntesi que havia obert la 15 en relació a la dualitat d'ordres ***TEXTO*** i ***TEXTODIN***, de la qual ens fèiem ressò en començar a desenvolupar **C:GINSERT**, tot introduint el paper de la variable ***TEXTEVAL*** en l'execució de ***-INSERT***, sota el control directe de l'Editor de Dibuix o des d'AutoLISP. Dèiem que, sota el primer règim, ***TEXTO*** havia esdevingut sinònim de ***TEXTODIN***, i que només des de **command** la primera ordre tornava a diferenciar-se de la segona. Doncs bé, en règim interactiu, ACAD 2004 ha recuperat l'antic "modus operandi" de ***TEXTO*** amb la nova ordre ***-TEXTO***, mentre que ***TEXTO*** i ***TEXTODIN*** segueixen sent sinònims; pel que fa al seu ús des d'AutoLISP, es manté un comportament que ara s'inscriu de ple en la norma que enunciàvem (tot i que les diferències entre ***-TEXTO*** i ***TEXTO***, com entre ***-ATREDIT*** i ***ATREDIT***, van més enllà del marc formal de diàleg amb l'usuari), perquè **(command "-TEXTO" ...)** i **(command "TEXTO" ...)** es comporten com l'ordre ***-TEXTO***; i cal seguir fent **(command "TEXTODIN" ...)** per executar ***TEXTO*** o ***TEXTODIN*** (l'última expressió no necessita el concurs de cap argument per a la introducció del text).


Consumidors compulsius de tot el món, uniu-vos!
Consumidors compulsius de tot el món, uniu-vos!
Consumidors compulsius de tot el món, uniu-vos!
Consumidors compulsius de tot el món, uniu-vos!
Consumidors compulsius de tot el món, uniu-vos!
Consumidors compulsius de tot el món, uniu-vos!
Consumidors compulsius de tot el món, uniu-vos!
Consumidors compulsius de tot el món, uniu-vos!
Consumidors compulsius de tot el món, uniu-vos!

NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS

En aquest capítol i en els que segueixen donarem un altre pas: no ens limitarem a millorar els recursos existents, substituint determinades ordres bàsiques per [o complementant-les amb] altres de noves, com fèiem amb ***GINSERT*** o ***RATREDIT***, sinó que en crearem un d'inèdit. En concret, no és pas que la nova ordre que prepararem parteixi de zero: segueix articulant-se a l'entorn de l'objecte compost anomenat bloc i fa ús de les ordres ***-BLOQUE*** i ***-INSERT*** (que, com assenyalàvem en acabar el capítol precedent, des de **command** poden ser invocades com a ***BLOQUE*** i ***INSERT***), però a diferència del que havíem estat fent (estaviar-li a l'usuari la feina d'executar explícitament diverses ordres o de reiterar ínputs), ara ens marquem com objectiu una fita impossible d'assolir mitjançant una seqüència d'ordres bàsiques.

En essència, el que volem és estendre la tècnica de les insercions de bloc al camp de les tranformacions afins, definides no des de l'òptica analítica de reducció a accions simples (translacions, girs i escalats) convenientment concatenades, sinó des d'una visió finalista, definint-les indirectament a partir d'una col·lecció de posicions i de les seves transformades: 3 punts no alineats (en 2D) o 4 punts no coplanaris (en 3D). Si la cosa anés d'obtenir els paràmetres de transformació per aplicar-la a tots els punts definitoris de la geometria d'una figura, tot plegat es reduiria a un problema algebraic: per a qui se'n pugui aprofitar, al final hem afegit un ANNEX: ENFOC ALGEBRAIC DE LES TRANSFORMACIONS AFINS. Però no és aquest el tema, sinó la consecució d'un sistema que assoleixi el mateix objectiu partint de l'especificitat de la base de dades d'AutoCAD i de les peculiaritats i regles de joc de l'objecte *BLOQUE*.

Tot i que l'anecdotari personal o la confessió de motivacions íntimes acostumen a ser perspectives poc habituals (i gens "correctes políticament", per usar la pobra terminologia filla del Pensament Únic) en la justificació de treballs de recerca aplicada, des de l'il·lusori semianonimat que proporciona el plural impersonal (que no majestàtic) entenem que pot tenir interès revelar quines han estat les idees-força d'on ha sortit **C:INSERTOK** (o ***INSERTOK***, si la tractem com una ordre AutoCAD), que així anomenarem l'aplicació AutoLISP que ens ocuparà a partir d'ara:

- L'experiència, mil cops repetida per qualsevol docent d'AutoCAD, que la inserció d'un bloc quadrat ortogonal és rectangular però que, si el bloc quadrat no és ortogonal (perquè hem girat el quadrat abans de convertir-lo amb bloc o perquè inserim un bloc compost per la inserció girada d'un bloc ortogonal), el resultat serà un paral·lelogram oblic (fora del cas particular d'escalat uniforme). Però, així com és factible, dominant l'ordre ***INSERT*** o usant aplicacions tan senzilles com ***GINSERT***, inserir un bloc quadrat ortogonal **1 × 1** de manera que s'emmotlli perfectament a un rectangle prefixat, no hi haurà manera d'adaptar la inserció d'un quadrat no ortogonal **1 × 1** a un paral·lelogram oblic prefixat, entre altres coses pel desconeixement *a priori* de quina inclinació hauria de tenir en origen.
- La constatació que la construcció geomètrica anomenada de Ritz, habitualment utilitzada per obtenir els diàmetres principals d'una el·lipse (orientació i longitud) a partir d'un parell de diàmetres conjugats, podia ser la clau del problema, considerant que l'eix principal major és l'eix d'una homologia afí de direcció ortogonal (a partir d'ara ens limitarem a parlar de "afinitat") entre aquesta el·lipse i una circumferència, i que els homològics dels dos diàmetres conjugats de l'el·lipse en la circumferència són perpendiculars.

El tema que abordem té una certa similitud amb el que justificava la creació de **C:GINSERT**, però va molt més enllà. En tots dos casos partim d'un bloc ortogonal unitari: no cal que sigui un quadrat de coordenades **0,0** (punt base per a les insercions), **1,0**, **1,1** i **0,1**, però si llavors calia que el punt **1,1** fos clarament identificable (en el sentit de poder decidir on volíem situar-lo en inserir el bloc), ara haurem de poder referir-nos als punts **1,0** i **0,1**. Quant al problema que s'ha de resoldre, ja no és el d'encabir el quadrat ortogonal en un rectangle de dimensions i orientació qualssevol, sinó en un paral·lelogram. No pas realitzant una inserció simple, que això és impossible, sinó inserint el quadrat ortogonal amb una determinada inclinació, convertint aquesta inserció en un nou bloc i inserint-lo, al seu torn, amb uns determinats factors d'escala. O, si ho voleu dir de una altra manera, realitzant la inserció d'un quadrat **1 × 1**, però no definit en posició ortogonal sinó girada. Al cap i a la fi estem parlant de la mateixa cosa (en comptes d'inserir un bloc constituït per la inserció prèvia del bloc ortogonal

podríem haver-la explosionat i ens trobaríem en el segon supòsit), i només es una qüestió de rendibilitat preferir el primer procediment al segon: si pensem que el bloc no té perquè ser estrictament un quadrat (pot ser una figura molt complexa, integrada per un gran nombre d'objectes AutoCAD, sempre que hi puguem identificar un quadrat unitari per usar-lo com a referència) i que, en general, l'encaix en paral·lelograms diferents requerirà partir de quadrats amb la orientació també diferent, sembla força més raonable disposar d'un únic bloc original, d'orientació ortogonal (només en aquest bloc caldrà describir exhaustivament el contingut), i definir blocs secundaris, un per a cada subconjunt de paral·lelograms susceptibles d'acollir-ne insercions, de contingut exigu (únicament les referències al bloc original, és a dir, un nom, tres coordenades, tres factors d'escala i un angle); altrament, cada bloc secundari contindria un volum d'informació explícita igual al de l'original (una descripció exhaustiva dels seus components) i no considerem que sigui admissible tanta redundància.

El problema geomètric l'haurem de resoldre en dues etapes:
- El gir que s'ha de donar a la inserció del bloc primitiu per dónar lloc als diferents blocs secundaris, cadascun dels quals es constituirà en matèria primera per al subconjunt d'insercions corresponent a tota una sèrie de paral·lelograms d'encaix.
- Els factors d'escala amb què cal inserir cada cop un determinat bloc secundari, que anomenarem genèric en relació al subconjunt de paral·lelograms associat, per assolir l'encaix desitjat.

Però, paral·lelament, caldrà abordar qüestions de procediment que tenen a veure amb un tema tan fonamental com aquest: una vegada haguem creat un bloc secundari, genèric d'un determinat subconjunt d'insercions, i arribi el moment de dur a terme més d'aquestes insercions, caldrà poder detectar-ne l'existència per evitar la proliferació de blocs secundaris equivalents i l'augment de la redundància del sistema. Aquesta qüestió, però, en suscita dues més:
- Com jugar amb els factors d'escala, per tal que els blocs secundaris siguin realment genèrics i de fàcil localització i manipulació. Concretant, ¿què és millor, usar $\mathbf{E_{1x} = E_{1y} = 1}$ en inserir el bloc original per donar lloc a cada genèric, encara que en la primera inserció d'aquest (i en totes les que hagin de seguir) sigui $\mathbf{E_{2x} \neq 1}$, o usar $\mathbf{E_{1x} = E_{1y} \neq 1}$ en inserir el bloc original, per tal que almenys en la primera inserció pugui ser $\mathbf{E_{2x} = 1}$? En el moment oportú ja veurem que ni una cosa ni l'altra, sinó que hi ha una tercera vía més pràctica.
- Com batejar aquests blocs genèrics, per tal que la informació geomètrica específica es trobi continguda en el mateix nom i no calgui emmagatzemar-la en la base de dades (annexada al bloc com a atributs invisibles o com a extensió de dades) o en arxius complementaris. En relació amb aquest tema cal no descuidar la manera d'organitzar l'estructura de blocs: en 2D, bloc original **BLOC** i blocs genèrics de primer grau, **BLOC*** (que contenen una inserció girada de **BLOC**, amb $\mathbf{E_{1x} = E_{1y} = E_{1z}}$), per inserir-los girats amb $\mathbf{E_{2x} \neq E_{2y}}$ i $\mathbf{E_{2z} = 1}$; en 3D, bloc original **BLOC**, blocs genèrics de primer grau, **BLOC*** (que contenen una inserció girada de **BLOC**, amb $\mathbf{E_{1x} = E_{1y} = E_{1z}}$), i blocs genèrics de segon grau, **BLOC**** (que contenen una inserció girada de **BLOC***, amb $\mathbf{E_{2x} = E_{2y} \neq E_{2z}}$), per inserir-los girats amb $\mathbf{E_{3x} \neq E_{3y}}$ i $\mathbf{E_{3z} = 1}$.

Per optimitzar el sistema caldrà que el conjunt dels blocs genèrics de primer grau (**BLOC***) sigui el més reduït possible: en 2D seria millor, si **BLOC** té atributs no constants ni predefinits, que els seus valors no fossin definitoris dels **BLOC***s (no repercutissin en els seus noms); i en 3D, a més, que els **BLOC***s estiguessin lligats al menor nombre possible de paràmetres (és a dir, que quan més genèrics, millor), fins i tot a costa d'augmentar l'especialització (l'especificitat) de **BLOC**** i el nombre d'elements de cada subconjunt **BLOC**** subsidiari d'un **BLOC***.

A la Figura 8.1 es fa palès com la tendència a ajornar la informació específica, desplaçant-la cap a nivells més pròxims a les insercions finals (en paral·lel a un procés d'ampliació del perfil dels blocs a cada nivell), com en les solucions de la dreta, redunda en un sistema menys redundant (valgui la redundància!), no sols perquè aconseguim articular-ho tot amb un menor (si més no, no major) nombre de blocs en cadascun dels nivells (i en total, com és lògic), sinó perquè els blocs d'un mateix nivell contenen menys quantitat d'informació: en l'exemple 2D tenim a la dreta 2 genèrics **BLOC*** (3 blocs en total) en lloc dels 4 de l'esquerra (5 blocs en total), per al mateix nombre d'insercions dobles (8); en l'exemple 3D tenim a la dreta 2 genèrics **BLOC*** i 4 **BLOC**** (7 blocs en total) en comptes dels 4 **BLOC***s i 8 **BLOC****s de l'esquerra (13 blocs en total), per a una mateixa quantitat d'insercions triples (16). Com que en els capítols següents ja hi tornarem, a propòsit de les opcions que poden afavorir aquesta tendència, canviem de tema.

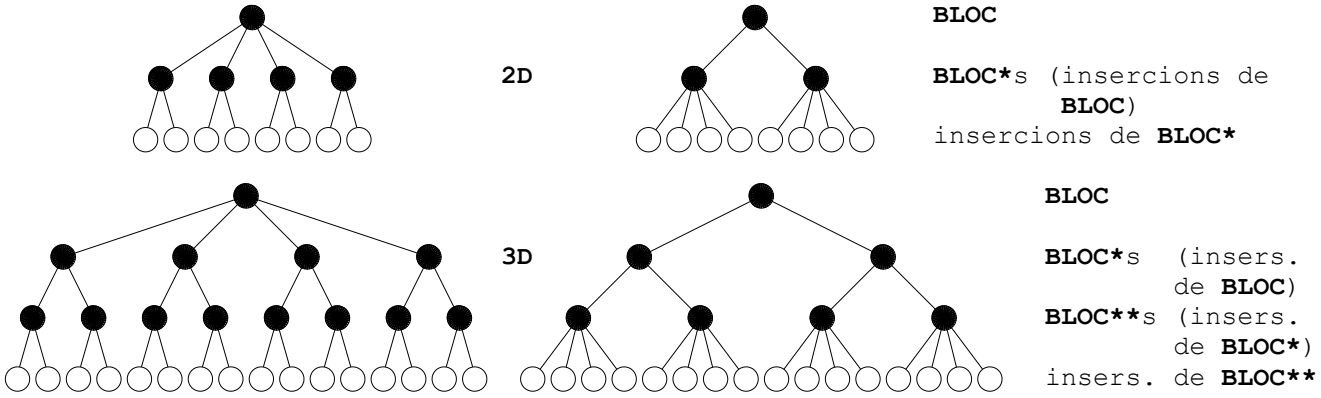


Figura 8.1

Centrant-nos en l'àmbit 2D, al qual ens limitarem en aquest capítol (en què encara no considerarem la presència d'atributs) i el següent (on ja els implementarem), comencarem per l'anàlisi geomètrica de la transformació, de què depenen les demés qüestions: determinarem el gir amb què s'ha d'inserir **BLOC**, i el gir i la relació d'escales $\frac{\mathbf{E_y}}{\mathbf{E_x}}$ amb què cal inserir aquesta inserció (no en diguem **BLOC***, perquè pel mig encara hi queda l'escalat uniforme de conveniència a què fa poc ens referíem), per adaptar al paral·lelogram d'acollida (**O** punt d'inserció, **X** transformat de **1,0** i **Y** transformat de **0,1**) el quadrat unitari ortogonal que suposem associat a **BLOC**. Així doncs, en lloc de considerar que $\mathbf{E_{1x} = E_{1y}}$ són els factors d'escala amb què de debò inserirem **BLOC** per constituir **BLOC*** i que $\mathbf{E_{2x} \neq E_{2y}}$ són els que acabarem utilitzant en la inserció d'aquest bloc genèric, per situar-nos en unes condicions en què ja es doni una relació d'afinitat entre quadrat i paral·lelogram, partirem d'una mena de **BLOC*** provisional en què el gir α_1 de **BLOC** sigui el definitiu i que resulti d'un escalat uniforme amb uns factors $\mathbf{E_{1x}} = \mathbf{E_{1y}}$ tals que, quan l'anem a inserir en el vèrtex **O** del paral·lelogram **X–O–Y**, només resti girar-lo un angle α_2 (el quadrat farà $\mathbf{E_{1x} \times E_{1y}}$ i se situarà a **X′–O–Y′**) i escalar-lo $\mathbf{E_{2y}}$ ($\left|\mathbf{E_{2y}}\right| \neq \mathbf{E_{2x} = 1}$) en la direcció $\mathbf{X–X' \equiv Y–Y'}$ per deixar-lo encaixat: la construcció de Ritz permet determinar el quadrat a partir del paral·lelogram, i així avaluarem α_1, $\mathbf{E_{1x}} = \mathbf{E_{1y}}$, α_2 i $\mathbf{E_{2y}}$ (adoneu-vos que subíndexs i majúscules responen a nomenclàtors diferents i que, en relació a **BLOC**, $\mathbf{E_x = E_{1x}\,E_{2x} = E_{1x}}$ i $\mathbf{E_y = E_{1y}\,E_{2y} = E_{1x}\,E_{2y}}$). La Figura 8.2 mostra dos casos, $\left|\mathbf{E_{2y}}\right| < \mathbf{1}$ (esquerra) i $\left|\mathbf{E_{2y}}\right| > \mathbf{1}$ (dreta), en què l'eix d'afinitat és **X″–O–Y″**, l'angle α_1 és **X″–O–X′** (la inclinació de **O–X′** respecte a l'eix **O–X″**) i l'angle α_2 és **X–O–X″** (la inclinació de l'eix **O–X″** respecte a **O–X**), i on la suma $\gamma = \alpha_1 + \alpha_2$ representa l'angle **X–O–X′** (la inclinació de **O–X′** respecte a **O–X**).

Per simplificar els dibuixos, a la figura no es representen sencers el quadrats ni els paral·lelograms, sinó només els parells de costats concurrents **X′–O–Y′** i **X–O–Y**. Començarem girant **Y** $\pm 90°$ al voltant de **O** (per exemple, en el sentit que el deixi més a prop de **X**) i anomenarem **A** aquesta posició. Unirem **A** amb **X** i trobarem el punt mig **B** d'aquest segment. Amb centre a **B**, dibuixarem una circumferència que passi per **O**, que tallarà la recta determinada per **A** i **X** en els punts **C** i **D**. Per ser **C–D** un diàmetre i **O** un punt de la circumferència, **O–C** i **O–D** seran perpendiculars i demostrarem que representen la direcció i l'eix d'afinitat: prolongant el radi **O–B** fins que talli en **X′** una recta paral·lela a **O–C** per **X**, **O–X′** serà el costat del quadrat homòleg de **O–X**; dibuixant una perpendicular a **O–X′** per **O**, fins que talli una recta paral·lela a **O–C** per **Y**, **O–Y′** serà el costat homòleg de **O–Y**. Cal precisar que en la construcció de Ritz els punts **C** i **X** sempre se situen a bandes oposades respecte a **B** (de manera que sigui $\left|\mathbf{E_{2y}}\right| < \mathbf{1}$), mentre que nosaltres només ho fem aixi quan l'angle **X–O–Y** és obtús i col·loquem **C** i **X** a la mateixa banda de **B** (de manera que sigui $\left|\mathbf{E_{2y}}\right| < \mathbf{1}$) quan l'angle és agut. La decisió no és banal: com veurem de seguida, ens permetrà de reduir a la meitat el nombre de genèrics **BLOC***.

Si anomenem **X″** i **Y″** els peus de les perpendiculars a l'eix d'afinitat llançades des de **X** i **Y** respectivament, les demostracions que **O–Y′ = O–X′** (1ª part) i que $\frac{\mathbf{X''-X}}{\mathbf{X''-X'}} = \frac{\mathbf{Y''-Y}}{\mathbf{Y''-Y'}} = \mathbf{E_{2y}}$ (2ª part) serveixen per a les dues situacions i són elementals:

1ª part:

1.1- Per ser **X–X′** paral·lel a **O–C**, els triangles **B–X–X′** i **B–C–O** són homotètics.

1.2- Com que **B–C = B–O**, **B–X = B–X′**; i com que **B–A = B–X**, **B–A = B–X′**.

1.3- Per ser també **B–O = B–D**, els triangles **B–A–X′** i **B–D–O** són homotètics, i **A–X′** i **O–D** paral·lels.

1.4- Prolongant **X′–A** fins a tallar **O–C** en **E**, per ser perpendiculars **O–Y″** i **O–E**, **O–Y** i **O–A**, **O–Y′** i **O–X′**, i **Y″–Y–Y′** i **E–A–X**, els triangles **O–Y″–Y** i **O–E–A** són semblants, i els **O–Y–Y′** i **O–A–X′** també.

1.5- Però no només semblants: per ser **O–Y = O–A**, aquests triangles són congruents (iguals). I, òbviament, també els triangles **O–Y″–Y′** i **O–E–X′**.

1.6- En conseqüència, **O–Y″ = O–E = X″–X′**, **Y″–Y = E–A**, **Y–Y′ = A–X′**, **Y″–Y′ = E–X′** i **O–Y′ = O–X′**.

2ª part:

2.1- Els triangles **C–E–A** i **X–X″–D** no només són semblants, per ser paral·lels **E–C** i **X″–X**, **E–A** i **X″–D**, i **C–A** i **X–D**: com que **C–A = (C–B ∓ A–B) = (B–D ∓ B–X) = X–D**, també són congruents.

2.2- Per ser **E–A** i **X″–D** paral·lels i d'igual longitud, **E–X″** és paral·lel a **X–D**. I, com que **X″–X** i **X″–X′** estan alineats, els triangles **X–X″–D** i **X″–X′–E** són semblants, amb una raó de semblança $\frac{\mathbf{X-X''}}{\mathbf{X''-X'}} = \frac{\mathbf{E-D}}{\mathbf{E-X'}}$.

2.3- Però **E–D = E–A = Y″–Y** i **E–X′ = Y″–Y′**. En conseqüència, $\frac{\mathbf{X''-X}}{\mathbf{X''-X'}} = \frac{\mathbf{Y''-Y}}{\mathbf{Y''-Y'}}$.

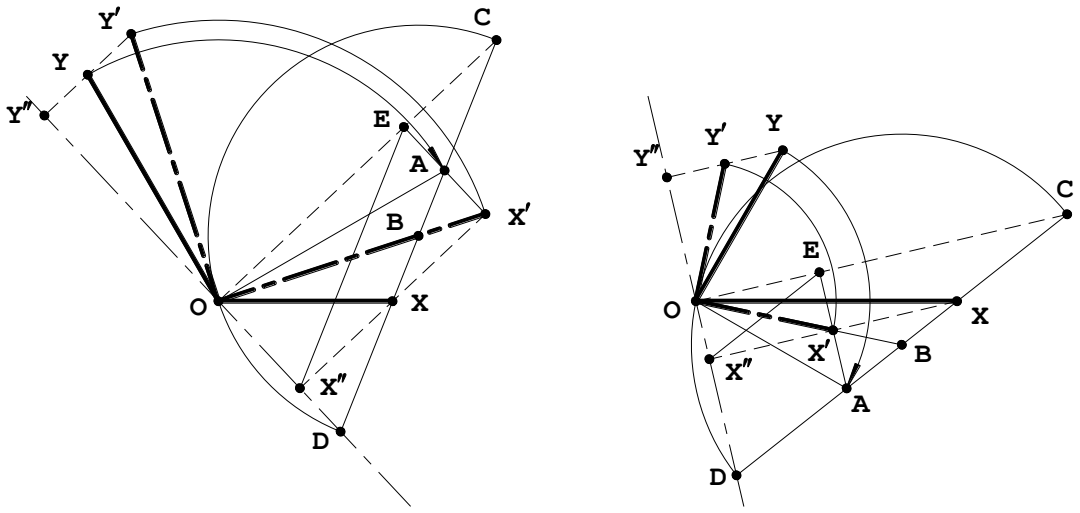


Figura 8.2

Si volem avaluar la versatilitat d'un bloc construït així, és a dir, la quantitat de variacions del paral·lelogram d'acollida (permutació de **X** i **Y**; permutació dels costats que comparteixen **O**; substitució de l'angle pel seu suplementari) que pot suportar el **BLOC*** previst per a la forma canònica o, si us estimeu més l'enunciat recíproc, quants **BLOCS***s es necessiten per completar les 8 assignacions possibles de punts **X–O–Y** a vèrtexs consecutius d'un mateix paral·lelogram (si **BLOC** fos una figura simètrica, serien les 4 resultants d'adaptar primer **X–O–Y** i després **Y–O–X** a dos vèrtexs **O** consecutius, però en una figura qualsevol cal estendre l'adaptació a tots quatre vèrtexs), començarem per analitzar la resposta d'un bloc obtingut aplicant estrictament la construcció de Ritz, amb el benentès que ara parlem dels genèrics **BLOC*** de debò (no pas dels restrictius **BLOC***s provisionals, exclusivament concebuts per a factors $\mathbf{E_{2x} = 1 \neq \left|E_{2y}\right|}$ tals que el bloc i la inserció fossin afins geomètricament), perquè se'ls ha de poder aplicar qualsevol factor $\mathbf{E_{2x} > 0}$ i $\mathbf{E_{2y}}$.

A més, cal puntualitzar què entenem per escalat uniforme: quan afirmàvem que havia de ser $\mathbf{E_{1x}} = \mathbf{E_{1y}}$, hauríem hagut d'ampliar la definició a $\mathbf{E_{1x}} = \left|\mathbf{E_{1y}}\right|$ i admetre una simetria implícita. Aclarit això, considerem els 8 casos d'encaix que ens mostra la Figura 8.3, on ens hem limitat a considerar un **BLOC** quadrat amb una "P" (lletra asimètrica): les construccions que vèiem a la Figura 8.2 les hem girat de manera que l'eix d'afinitat sempre aparegués horitzontal, i s'hi representen els **BLOC***s de debò (resultants d'uns factors $\mathbf{E_{1x}}$ i $\mathbf{E_{1y}}$ que justificarem més endavant), els provisionals que havíem usat de primer i les insercions finals, afins amb aquests.

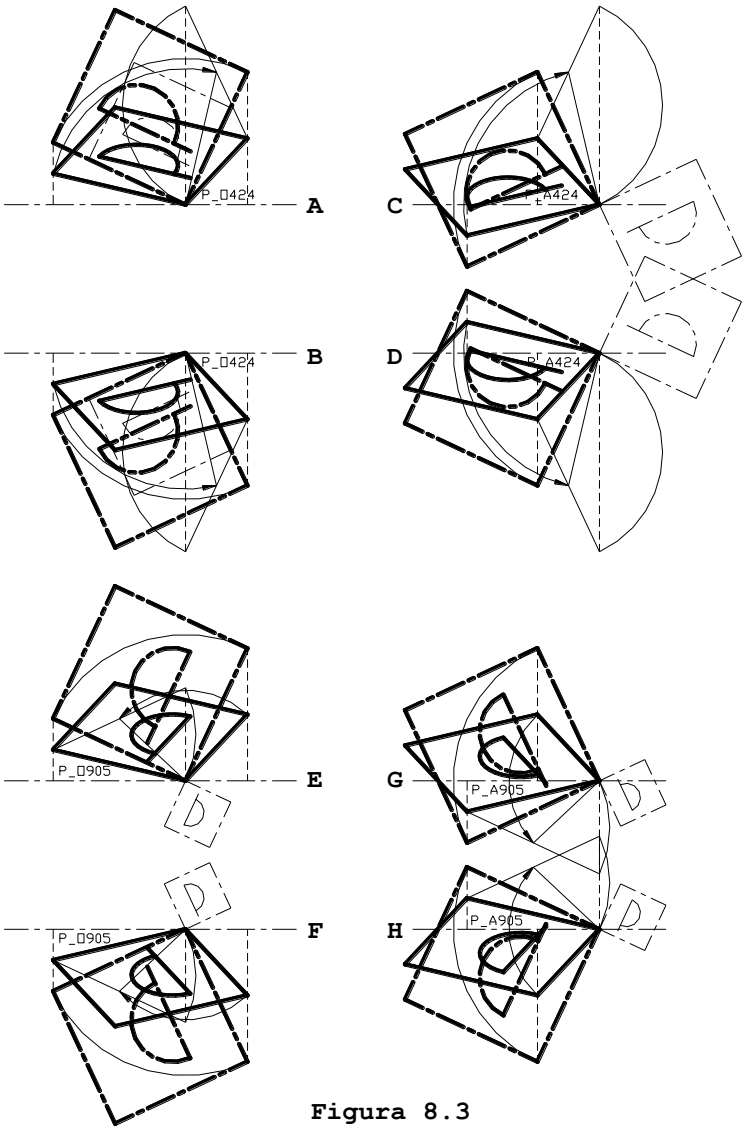


Figura 8.3

Els blocs A i B són simètrics respecte a **X** i intercanviables en el sentit que, si amb $\mathbf{0 < E_{2y} < 1}$ el bloc A s'insereix adaptant-se al seu paral·lelogram i el bloc B al seu, adoptant un factor $\mathbf{E'_{2y} = -E_{2y}}$ el bloc A s'adaptarà al paral·lelogram de B i el bloc B al de A. Que sigui la inserció de **BLOC** del cas A o la del cas B a qui acabem atorgant la categoria de genèric **BLOC*** només depèn de quin dels dos casos es produirà abans. La cosa hauria d'anar així: si, per exemple, primer s'esdevé el cas A i definim el genèric **BLOC*** amb la inserció de **BLOC** a què dóna lloc (amb $\mathbf{0 < \alpha_{a1} < 90°}$ i $\mathbf{E_{a1x} = E_{a1y} > 0}$), no només quan es repeteixi la situació sinó quan s'esdevingui el cas B i **C:INSERTOK** determini les característiques que hauria de tenir un genèric a mida ($\mathbf{\alpha_{b1} = -\alpha_{a1}}$, $\mathbf{E_{b1x} = E_{a1x}}$ i $\mathbf{E_{b1y} = -E_{a1y}}$) hauria de detectar l'existència del primer i saber que amb aquest **BLOC*** ja en tenim prou, perquè pot assumir perfectament les funcions del segon; i, si no volem que hagi de rastrejar en les definicions de tots els blocs existents, caldrà que en el nom de **BLOC*** hi hagi alguna referència al nom de **BLOC** i a aquelles característiques geomètriques, comunes als casos A i B, que no es donen en cap cas més.

Però, ¿quines són aquestes característiques que constitueixen a la vegada la marca d'identificació de cada genèric i la diferència específica respecte als demés? ¿n'hi ha prou amb l'angle de gir? Només amb l'angle, no: d'acord que hem parlat de l'equivalència funcional de les insercions de **BLOC** simétriques respecte l'eix **X** (en el sentit de ser intercanviables), però en termes de paràmetres d'inserció cal tenir en compte que aquesta simetria comporta $\mathbf{E_{j1x} = E_{i1x}}$ i $\mathbf{E_{j1y} = -E_{i1y}}$ però també $\alpha_{j1} = -\alpha_{i1}$, i això només s'assoleix entre els casos A i B, com hem vist, i entre C i D ($\alpha_{c1} = \alpha_{b1} = -\alpha_{a1} = -\alpha_{d1}$, $\mathbf{E_{c1x} = E_{d1x} = E_{a1x} = E_{b1x}}$ i $\mathbf{E_{c1y} = E_{a1y} = -E_{b1y} = -E_{d1y}}$), però no entre A i C o A i D, ni entre B i C o B i D, de manera que als dos últims casos els cal un genèric **BLOC*** diferent (podrà ser el propi de C o el propi de D). Anàlogament, els casos E, F, G i H, on **X** i **Y** permuten posicions (i en què $|\alpha_{e1}| = \ldots = |\alpha_{h1}| \neq |\alpha_{a1}| = \ldots = |\alpha_{d1}|$, $\mathbf{E_{e1x} = \ldots = E_{h1x} \neq E_{a1x} = \ldots = E_{d1x}}$ i $|\mathbf{E_{e1y}}| = \ldots = |\mathbf{E_{h1y}}| \neq |\mathbf{E_{a1y}}| = \ldots = |\mathbf{E_{d1y}}|$, tret que el paral·lelogram fos un rombe) precisarem dos genèrics més: un per E i F, i un altre per G i H. Com veieu, no és tant que el signe de l'angle sigui un destorb (si fos així, podríem adoptar com identificador el valor $|\alpha_1|$ o $\cos\alpha_1$, sense més) com que hi ha un altre element significatiu: la coincidència o no entre aquest signe i el de $\mathbf{E_{1y}}$. Així, per diferenciar entre el genèric de A+B i el de C+D, i entre el de E+F i el de G+H, podríem incorporar els signes "**+**" o "**-**", assignant "**+**" quan α_1 i $\mathbf{E_{1y}}$ tinguessin el mateix signe i "**-**" quan el tinguessin oposat, però com que ja havíem pensat en el caràcter "**_**" per a la composició del nom de **BLOC***, lligant el nom de **BLOC** a l'identificador geomètric que discutim, el risc de confusió entre "**_**" i "**-**" ens ha fet decantar per una "**O**" en el primer cas (angle **X–O–Y** obtús) i una "**A**" en el segon (angle **X–O–Y** agut). Com que la qualificació de l'angle **X–O–Y** (agut, recte, obtús), prèvia a la funció **INSERT*** (que és on es realitzarà virtualment la construcció de Ritz, es compondrà el nom **BLOC***, es crearà aquest genèric si encara no existia i s'inserirà de manera que el quadrat ortonormal de referència encaixi en el paral·lelogram **X–O–Y**), és una operació amb què ja hi comptàvem, precisament per determinar si cal accedir a **INSERT*** o el cas es pot despatxar amb una inserció simple de **BLOC** (encara que el recurs a **C:INSERTOK** només està justificat quan el paral.lelogram d'encaix no és un rectangle, pot succeir que la precisió de càlcul –representada per la constant **Q0**– no permeti filar massa prim i assimili a rectangles alguns paral·lelograms de poca obliqüitat o, simplement, que l'usuari no se n'hagi adonat), l'aprofitarem en la composició del nom **BLOC***: concatenarem el nom **BLOC** amb "**_A**" o "**_O**" i, pel que fa al paràmetre numèric, ens quedarem amb la part decimal de $\cos\alpha_1 = \dfrac{\mathbf{O–X''}}{\mathbf{O–X'}}$.

La traducció a codi de tot això (incloent-hi els factors $\mathbf{E_{1x}} = \dfrac{\mathbf{O–X'}}{\mathbf{O–X''}}$ i $\mathbf{E_{1y}} = \pm\dfrac{\mathbf{O–X'}}{\mathbf{O–X''}}$ que més endavant justificarem), és:

```
(defun INSERT* (/ X* X** Y** OX* OX** BLOC*)
  (setq A (if (< I 0) A B)
        W (angle X A)
        B (polar X W (/ (distance X A) 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC* (strcat BLOC "_" (if (< J K) "A" "O")
                      (itoa (fix (/ OX** OX* Q0))))
        J 1 K (tblsearch "BLOCK" BLOC*))
  (if K
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 K))))) 0) -1 1)))
      (command "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*
               "SCP" "Z" (setq K (trans O 1 0) O O) X**
               "BLOQUE" BLOC* (trans K 0 1) "LT" ""
               "SCP" "PR"))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))
```

```
(defun C:INSERTOK (/ Q0 PI/2 ECO OSN BLOC O X Y OX OY XY A B E LE W I J K)
  (setq Q0 0.001
        PI/2 (/ PI 2)
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        BLOC (getstring "\nIndique nombre de bloque: " T)
        O (getpoint "\nPunt d'insercio: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC  " <1>: "))
              W (angle O A)
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
  (if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
      (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
          (command "INSERT" BLOC O OX (* OY I) X)
          (INSERT*)))
  (setvar "INSNAME" BLOC)
  (setvar "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))
```

Quin és el defecte d'aquesta primera aproximació? Sens dubte, la limitació que abans havíem expressat en la forma $|E_{2y}| < 1$, on el factor correspon a la inserció del **BLOC*** provisional, perquè si utilitzem els d'una inserció del **BLOC*** definitiu hauríem de dir que $\frac{|E_{2y}|}{E_{2x}} < 1$. Tant en un cas com en l'altre, és $\frac{X''-X}{X''-X'} = \frac{Y''-Y}{Y''-Y'} < 1$: en el primer per definició de E_{2y} i en el segon perquè, a partir dels valors E_{1x} i E_{1y} adoptats en **INSERT*** (pendents de justificació, recordem-ho), $E_{2x} = O-X'' = Y''-Y'$ (per igualtat dels triangles $X'-X''-O$ i $O-Y''-Y'$). Així, si considerem en el cas A ($X-O-Y$ obtús), per exemple, que $\frac{X''-X}{X''-X'} = \frac{Y''-Y}{Y''-Y'}$ va creixent, quan arribi a **1** ($X-O-Y$ recte) podrem resoldre l'encaix amb una inserció simple de **BLOC**. A partir d'aquí, si aquest valor seguiex creixent ($X-O-Y$ agut) passarem al cas H perquè, tret que fos un rombe ($\alpha_1 = 45° \rightarrow X''-X' = O-X'' = O-Y'' = Y''-Y' \rightarrow X-O-Y$), $O-X < O-Y$ canviarà a $O-X > O-Y$: les regles de joc actuals no permeten que hi hagi continuïtat (interrompuda només per la singularitat $X = X'$, $Y = Y'$) sobre la base de seguir usant el mateix genèric P_O424 ($\cos 64{,}9° = 0{,}424$), com a la Figura 8.4, i haurem de recórrer al bloc genèric P_A905 ($\cos(64{,}9°-90°) = 0{,}905$). Dit de forma planera: si l'algorisme dissenyat sols té capacitat per aixafar el quadrat (tot comprimint la diagonal que parteix de **O**), per estirar-lo (tot allargant aquesta diagonal) l'haurà de girar $\pm 90°$, seguir aixafant-lo, si bé ara en direcció transversal (tot comprimint la diagonal $X-Y$), i aplicar un escalat uniforme suplementari (per restituir a aquesta segona diagonal la longitud original).

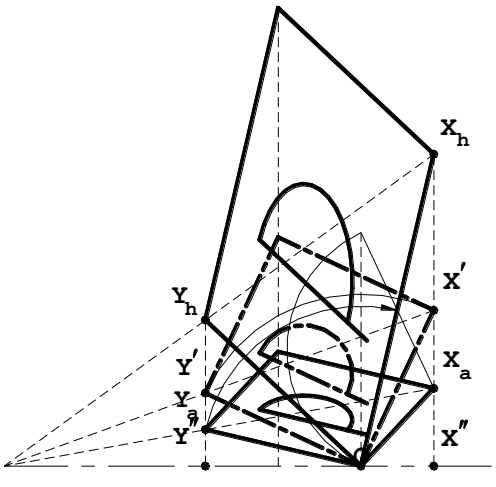


Figura 8.4

Tot canviarà radicalment, per obra i gràcia del nou criteri d'ubicació del punt **C**, en relació a **B** i **X**, en la construcció de Ritz: per a cada subconjunt d'adaptacions del bloc P a un determinat subconjunt de paral·lelograms geomètricament semblants, en lloc de 4 blocs derivats genèrics (P_O424, P_A424, P_O905 i P_A905) n'hi haurà prou amb 2 (P_424 i P_905), perquè l'ampliació de $\left|\mathbf{E_{2y}}\right| < 1$ a $\left|\mathbf{E_{2y}}\right| \neq 1$ comporta la dissolució de la frontera entre angles **X–O–Y** aguts i obtusos. A la Figura 8.5 hi tenim de nou els paral·lelograms d'encaix de la Figura 8.3, amb l'única diferència aparent de les posicions C, D, G i H (en què l'eix d'afinitat ha girat $\pm 90^{\circ}$ i, com que sempre el presentem horitzontal, hem hagut de girar $\mp 90^{\circ}$ tota la construcció), però, pel que fa a les definicions de **BLOC***, també hi apreciem diferències reals.

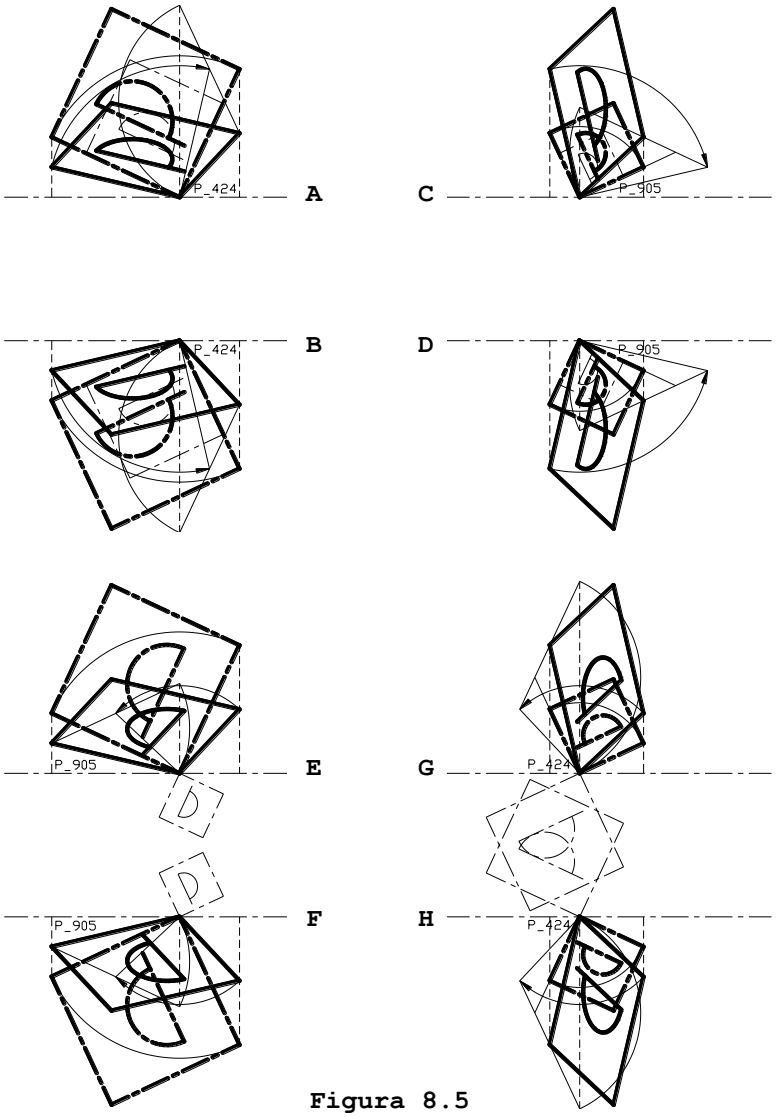


Figura 8.5

Igual que abans, els blocs A i B són simètrics respecte a **X** i intercanviables en el sentit que, si amb $\mathbf{0 < E_{2y} < 1}$ A s'insereix adaptant-se al seu paral·lelogram i el B al seu, adoptant un factor $\mathbf{E'_{2y} = -E_{2y}}$ A s'adaptarà al paral·lelogram de B i B al de A. També com abans, que sigui la inserció de **BLOC** del cas A o la del cas B a qui acabem atorgant la categoria de genèric **BLOC*** només dependrà de quin dels dos casos es produeix abans. Havíem vist que la especificitat de cada genèric residia a l'angle de gir però que, en atenció a l'equivalència funcional de les insercions de **BLOC** simétriques respecte l'eix **X** (en el sentit de ser intercanviables), calia tenir en compte que aquesta simetria comportava $\mathbf{E_{j1x} = E_{i1x}}$, $\mathbf{E_{j1y} = -E_{i1y}}$ i també $\alpha_{j1} = -\alpha_{i1}$. Com abans, això es dóna entre els casos A i B ($\mathbf{\alpha_{b1} = -\alpha_{a1}}$, $\mathbf{E_{b1x} = E_{a1x}}$ i $\mathbf{E_{b1y} = -E_{a1y}}$) i entre C i D ($\mathbf{\alpha_{d1} = -\alpha_{c1}}$, $\mathbf{E_{d1x} = E_{c1x}}$ i $\mathbf{E_{d1y} = -E_{c1y}}$), però no entre A i C o A i D, ni entre B i C o B i D ($\left|\mathbf{\alpha_{c1}}\right| \neq \left|\mathbf{\alpha_{a1}}\right|$, $\mathbf{E_{c1x} \neq E_{a1x}}$ i $\left|\mathbf{E_{c1y}}\right| \neq \left|\mathbf{E_{a1y}}\right|$), de manera que als dos últims casos els cal un genèric **BLOC*** diferent (podrà ser el

propi de C o el propi de D). Però aquí s'acaba l'analogia amb la situació mostrada a la Figura 8.3, perquè en els casos E, F, G i H, on **X** i **Y** permuten posicions, el bloc E és idèntic al D, el bloc F és idèntic al C (tots quatre casos constitueixen un mateix genèric **BLOC***), el bloc G és idèntic al B i el bloc H és idèntic al A (tots quatre casos constitueixen un altre genèric **BLOC***): a diferència d'abans, en què quan **X–O–Y** era obtús el signes de l'angle α_1 i del factor E_{1y} coincidien, i quan era agut tenien signe contrari, ara α_1 i E_{1y} tenen el mateix signe tant en el genèric A+B+G+H com en el genèric C+D+E+F, i tant l'un com l'altre tenen la capacitat d'adaptar-se als dos tipus d'obliqüitat, raó per la qual el nom **BLOC*** podrà desprendre's del distintiu "**A**/**O**" i limitar-se al paràmetre numèric (la part decimal de $\cos\alpha_1 = \frac{O–X''}{O–X'}$).

La traducció a codi de tot això (incloent-hi els factors $E_{1x} = \frac{O–X'}{O–X''}$ i $E_{1y} = \pm\frac{O–X'}{O–X''}$ que tot seguit justificarem) només afecta la funció **INSERT***, on hem subratllat les modificacions respecte a la versió precedent:

```
(defun INSERT* (/ X* X** Y** OX* OX** BLOC*)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC* (strcat BLOC " " (itoa (fix (/ OX** OX* Q0))))
        J 1 K (tblsearch "BLOCK" BLOC*))
  (if K
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 K))))) 0) -1 1)))
      (command "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*
               "SCP" "Z" (setq K (trans O 1 0) O O) X**
               "BLOQUE" BLOC* (trans K 0 1) "LT" ""
               "SCP" "PR"))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))
```

Havent optimitzat la definició de **BLOC*** (en reduir-ne l'especificitat i assegurar la cobertura d'un conjunt aleatori de casos amb un mínim nombre d'exemplars), li ha arribat el torn a la determinació de la mida ideal des d'un punt de vista de la pràctica de la inserció (en definitiva del factor $E_{1x} = E_{1y}$ amb que convé inserir **BLOC** per constituir **BLOC***). La primera vegada que calgui recórrer a un nou **BLOC***, definint-lo a partir d'una inserció de **BLOC** escalada uniformement i girada un angle $\alpha_1 = \arccos\frac{O–X_1''}{O–X_1'}$, el més immediat seria adoptar els factors d'escala que havíem adoptat provisionalment per situar-nos en les condicions de la Figura 8.2, $E_{11x} = \left|E_{11y}\right| = O–X_1'$ (mentre ens mantinguem en 2D, serà $E_{11z} = E_{11x}$ i $E_{12z} = E_{12x}$), de manera que el quadrat 1×1 que suposem associat a **BLOC** estigués representat a **BLOC*** per un quadrat $O–X_1' \times O–X_1'$ i que el seu encaix en el paral·lelogram $X_1–O–Y_1$ es reduís a una transformació d'escalat en la direcció **Y** (una afinitat d'eix **X**, com dèiem abans). Per tal d'establir una pauta fàcilment aplicable a d'altres casos, ho plantejarem amb caràcter general, com un escalat amb origen a **O** en què

X_1' ($O–X_1''$, $\pm X_1''–X_1'$) es transforma en X_1 ($O–X_1''$, $\pm X_1''–X_1$) i

Y_1' ($O–Y_1''$, $\pm Y_1''–Y_1'$) es transforma en Y_1 ($O–Y_1''$, $\pm Y_1''–Y_1$)

Els factors d'escala haurien de ser:

$$E_{12x} = \frac{O–X_1''}{O–X_1''} = \frac{O–Y_1''}{O–Y_1''} = 1$$

$$E_{12y} = \pm\frac{X_1''–X_1}{X_1''–X_1'} = \pm\frac{Y_1''–Y_1}{Y_1''–Y_1'}$$

Però això no seria gaire pràctic, pel que fa a l'aprofitament de **BLOC*** per a nous paral·lelograms $\mathbf{X_n{-}O{-}Y_n}$ ($\mathbf{X_n}$ i $\mathbf{Y_n}$ de coordenades ($\mathbf{O{-}X''_n}$, $\mathbf{\pm X''_n{-}X_n}$) i ($\mathbf{O{-}Y''_n}$, $\mathbf{\pm Y''_n{-}Y_n}$), respectivament) en què $\frac{\mathbf{O{-}X''_n}}{\mathbf{O{-}X'_n}} = \cos \boldsymbol{\alpha_n} = \cos \boldsymbol{\alpha_1} = \frac{\mathbf{O{-}X''_1}}{\mathbf{O{-}X'_1}}$, perquè

$$\mathbf{E_{n2x}} = \frac{\mathbf{O{-}X''_n}}{\mathbf{O{-}X''_1}} = \frac{\mathbf{O{-}Y''_n}}{\mathbf{O{-}Y''_1}} \quad \text{i}$$

$$\mathbf{E_{n2y}} = \pm \frac{\mathbf{X''_n{-}X_n}}{\mathbf{X''_1{-}X'_1}} = \pm \frac{\mathbf{Y''_n{-}Y_n}}{\mathbf{Y''_1{-}Y'_1}}$$

I no només perquè el càlcul d'aquests factors fos més o menys laboriós, sinó perquè ens obligaria a conservar com a constants els valors $\mathbf{O{-}X''_1}$ i $\mathbf{\pm X''_1{-}X_1}$ (o bé $\mathbf{O{-}Y''_1}$ i $\mathbf{\pm Y''_1{-}Y_1}$), per poder-los utilitzar en totes les insercions de **BLOC***. Allò que mai no farem, com és lògic, és crear un bloc *ad hoc*, de dimensions $\mathbf{O{-}X'_n \times O{-}X'_n}$, per a cada paral·lelogram $\mathbf{X_n{-}O{-}Y_n}$, en contradicció amb el principi d'economia de mitjans que asseguràvem defensar.

Així que, per aconseguir un **BLOC*** realment genèric i d'utilització pràctica en totes les insercions associades a un determinat valor $\frac{\mathbf{O{-}X''_n}}{\mathbf{O{-}X'_n}} = \frac{\mathbf{O{-}X''_1}}{\mathbf{O{-}X'_1}}$ (és a dir, que no depengui d'un emmagatzament addicional d'informació, més enllà de l'estricta definició d'inserció de **BLOC**), n'hi ha prou a mantenir les mides del bloc original (inserció de **BLOC** amb $\mathbf{E_{1x} = E_{1y} = E_{1z} = 1}$ i el gir esmentat). En funció del primer paral·lelogram d'acollida $\mathbf{X{-}O{-}Y}$, que posarà en marxa la creació d'un nou **BLOC***, les coordenades dels vèrtexs $\mathbf{X'_{1u}}$ i $\mathbf{Y'_{1u}}$ seran

$\mathbf{X'_{1u}}$ $\left(\frac{\mathbf{O{-}X''_1}}{\mathbf{O{-}X'_1}}, \pm \frac{\mathbf{X''_1{-}X'_1}}{\mathbf{O{-}X'_1}}\right)$ que cal transformar en $\mathbf{X_1}$ ($\mathbf{O{-}X''_1}$, $\mathbf{\pm X''_1{-}X_1}$) i

$\mathbf{Y'_{1u}}$ $\left(\frac{\mathbf{O{-}Y''_1}}{\mathbf{O{-}X'_1}}, \pm \frac{\mathbf{Y''_1{-}Y'_1}}{\mathbf{O{-}X'_1}}\right)$ que cal transformar en $\mathbf{Y_1}$ ($\mathbf{O{-}Y''_1}$, $\mathbf{\pm Y''_1{-}Y_1}$)

Els factors d'escala hauran de ser:

$$\mathbf{E_{12x}} = \frac{\frac{\mathbf{O{-}X''_1}}{\mathbf{O{-}X''_1}}}{\frac{}{\mathbf{O{-}X'_1}}} = \frac{\frac{\mathbf{O{-}Y''_1}}{\mathbf{O{-}Y''_1}}}{\frac{}{\mathbf{O{-}X'_1}}} = \mathbf{O{-}X'_1}$$

$$\mathbf{E_{12y}} = \frac{\pm \frac{\mathbf{X''_1{-}X_1}}{\mathbf{O{-}X'_1}}}{\pm \frac{\mathbf{X''_1{-}X'_1}}{\mathbf{O{-}X'_1}}} = \frac{\pm \frac{\mathbf{Y''_1{-}Y_1}}{\mathbf{O{-}X'_1}}}{\pm \frac{\mathbf{Y''_1{-}Y'_1}}{\mathbf{O{-}X'_1}}} = \mathbf{O{-}X'_1}\,\frac{\mathbf{Y''_1{-}Y_1}}{\mathbf{Y''_1{-}Y'}}$$

Com era de preveure, en ser **BLOC*** $\mathbf{O{-}X'_1}$ vegades més petit (més gran, si $\mathbf{O{-}X'_1 < 1}$) que el d'abans, els factors d'escala hauran de ser $\mathbf{O{-}X'_1}$ vegades més grans (més petits), però alguna cosa més ha canviat: per a insercions posteriors (encaix en paral·lelograms $\mathbf{X_n{-}O{-}Y_n}$, on $\mathbf{X_n}$ és ($\mathbf{O{-}X''_n}$, $\mathbf{\pm X''_n{-}X_n}$) i i $\mathbf{Y_n}$ és ($\mathbf{O{-}Y''_n}$, $\mathbf{\pm Y''_n{-}Y_n}$)),

$$\mathbf{E_{n2x}} = \frac{\frac{\mathbf{O{-}X''_n}}{\mathbf{O{-}X''_1}}}{\frac{}{\mathbf{O{-}X'_1}}} = \frac{\frac{\mathbf{O{-}X''_n}}{\mathbf{O{-}X''_n}}}{\frac{}{\mathbf{O{-}X'_n}}} = \mathbf{O{-}X'_n} \qquad \left(\frac{\mathbf{O{-}X''}}{\mathbf{O{-}X'}} \text{ és constant en cada genèric}\right)$$

$$\mathbf{E_{n2y}} = \frac{\pm \frac{\mathbf{Y''_n{-}Y_n}}{\mathbf{O{-}X'_1}}}{\pm \frac{\mathbf{Y''_1{-}Y'_1}}{\mathbf{O{-}X'_1}}} = \frac{\frac{\mathbf{Y''_n{-}Y_n}}{\mathbf{O{-}X''_1}}}{\frac{}{\mathbf{O{-}X'_1}}} = \frac{\frac{\mathbf{Y''_n{-}Y_n}}{\mathbf{O{-}X''_n}}}{\frac{}{\mathbf{O{-}X'_n}}} \quad (\mathbf{O{-}Y''{-}Y'} \text{ i } \mathbf{X'{-}X''{-}O} \text{ iguals} \rightarrow \mathbf{Y''{-}Y' = O{-}X''})$$

i ara sí que podem afirmar que un **BLOC*** de dimensions idèntiques a **BLOC** és genèric en el subconjunt de les insercions caracteritzades per un determinat valor $\frac{\mathbf{O{-}X''}}{\mathbf{O{-}X'}}$, i autosuficient en la mesura que per calcular els factors $\mathbf{E_{n2x}}$ i $\mathbf{E_{n2y}}$ en totes les insercions posteriors a la inaugural n'hi haurà prou amb els paràmetres obtinguts en el mateix procés que ens ha permès certificar la seva adscripció a l'esmentat subconjunt i la pertinència de la utilització de **BLOC***.

De tota manera, com que cada **BLOC*** es defineix un sol cop, amb motiu de la primera inserció que en justifica l'existència, però podem inserir-lo moltes vegades més, no ens donem per satisfets i proposem una versió alternativa, igualment genèrica i

autosuficient però de mida diferent a **BLOC**, als efectes d'alleugerir els reiterats càlculs de $\mathbf{E_{n2x}}$ i $\mathbf{E_{n2y}}$. Crearem un **BLOC*** en què les dimensions que corresponen al quadrat $\mathbf{1 \times 1}$ associat a **BLOC** seran $\frac{O-X'}{O-X''} \times \frac{O-X'}{O-X''}$, i els vèrtexs $\mathbf{X'_{1v}}$ i $\mathbf{Y'_{1v}}$ tindran les coordenades

$$\mathbf{X'_{1v}} \quad \left(\frac{O-X''_1}{O-X'_1} \times \frac{O-X'_1}{O-X''_1},\ \pm\frac{X''_1-X'_1}{O-X'_1} \times \frac{O-X'_1}{O-X''_1}\right) \equiv \left(1,\ \pm\frac{X''_1-X'_1}{O-X''_1}\right)$$

que cal transformar en $\mathbf{X_1}$ $(O-X''_1,\ \pm X''_1-X_1)$ i

$$\mathbf{Y'_{1v}} \quad \left(\frac{O-Y''_1}{O-X'_1} \times \frac{O-X'_1}{O-X''_1},\ \pm\frac{Y''_1-Y'_1}{O-X'_1} \times \frac{O-X'_1}{O-X''_1}\right) \equiv \left(\frac{O-Y''_1}{O-X''_1},\ \pm\frac{Y''_1-Y'_1}{O-X''_1}\right) \equiv \left(\frac{O-Y''_1}{O-X''_1},\ \pm\frac{O-X''_1}{O-X''_1}\right) \equiv$$

$$\equiv \left(\frac{O-Y''_1}{O-X''_1},\ \pm 1\right)$$

que cal transformar en $\mathbf{Y_1}$ $(O-Y''_1,\ \pm Y''_1-Y_1)$

Els factors d'escala hauran de ser:

$$\mathbf{E_{12x}} = \frac{O-X''_1}{1} = \frac{O-Y''_1}{\frac{O-Y''_1}{O-X''_1}} = O-X''_1$$

$$\mathbf{E_{12y}} = \frac{\pm X''_1-X_1}{\pm\frac{X''_1-X'_1}{O-X''_1}} = \frac{\pm Y''_1-Y_1}{\pm 1} = Y''_1-Y_1$$ (la igualtat del 4t membre amb el 3r és òbvia,

però amb el 2n ho és menys: $\frac{X''_1-X_1}{X''_1-X'_1} = \frac{Y''_1-Y_1}{Y''_1-Y'_1} \rightarrow \frac{\frac{X''_1-X_1}{O-X''_1}}{\frac{X''_1-X'_1}{O-X''_1}} = \frac{\frac{Y''_1-Y_1}{O-X''_1}}{\frac{Y''_1-Y'_1}{O-X''_1}} = \frac{Y''_1-Y_1}{Y''_1-Y'_1} = Y''_1-Y_1$)

No cal dir que, per a insercions posteriors (encaix en paral·lelograms $\mathbf{X_n-O-Y_n}$, on $\mathbf{X_n}$ i $\mathbf{Y_n}$ tinguin les coordenades $(O-X''_n,\ \pm X''_n-X_n)$ i $(O-Y''_n,\ \pm Y''_n-Y_n)$), aquest genèric seguirà sent autosuficient, perquè el càlcul dels factors d'escala que cal aplicar en cada ocasió

$$\mathbf{E_{n2x}} = \frac{O-X''_n}{1} = O-X''_n$$

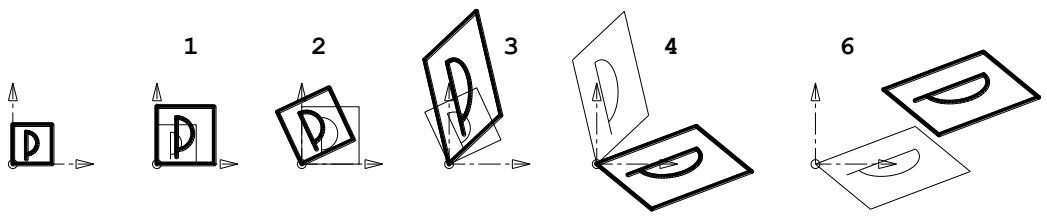
$$\mathbf{E_{n2y}} = \frac{\pm Y''_n-Y_n}{\pm 1} = Y''_n-Y_n$$

es farà a partir dels paràmetres obtinguts en el mateix procés i, pel que toca a $\mathbf{E_{n2y}}$, amb més simplicitat que amb un **BLOC*** no escalat.

Aclarida l'adopció dels factors $\mathbf{E_{11x}} = \left|\mathbf{E_{11y}}\right| = \frac{O-X'}{O-X''}$ en la inserció de **BLOC** qué dóna lloc a **BLOC***, abans de passar als aspectes no geomètrics de **C:INSERTOK**, seria bo recapitular, per entendre millor el procés des de l'òptica de les transformacions afins (enteses com a successió d'accions elementals, en sintonia amb l'exposició inicial de l'ANNEX: PLANTEJAMENT ALGEBRAIC DE LES TRANSFORMACIONS AFINS).

En intervenir l'efecte cisallament en la transformació quadrat → paral·lelogram, que constitueix el nucli dur de **C:INSERTOK**, i en respondre aquest efecte a una seqüència gir_1 + escalat_no_uniforme + gir_2, seria raonable identificar aquesta seqüència, entre d'altres accions, en la funció **INSERT***: tot plegat ho podríem esquematitzar en la Figura 8.6, on l'escalat $(\mathbf{E_x},\ \mathbf{E_y})$ es desdobla en dues fases ($\mathbf{E_{1x} = E_{1y} = E_x}$, $\mathbf{E_{2x} = 1}$ i $\mathbf{E_{2y} = \frac{E_y}{E_x}}$, de manera que $\mathbf{E_x = E_{1x}\ E_{2x}}$ i $\mathbf{E_y = E_{1y}\ E_{2y}}$), la primera de les quals la considerarem prèvia al gir (com que l'escalat és uniforme, tant se val escalar i girar com girar i escalar) i hi afegim una translació final.

Figura 8.6



De fet, el procediment seguit a **INSERT*** respon a aquesta estratègia en allò bàsic, però l'hem hagut de complicar una mica per tal d'adaptar-lo a les peculiaritats de l'objecte **"BLOCK"** d'AutoCAD (què fem quan definim un bloc i quan l'inserim): quan inserim un bloc assumint tots els valors per defecte (punt d'inserció $\mathbf{O \equiv (0,0,0)}$, $\mathbf{E_x = E_y = 1}$ i $\boldsymbol{\alpha = 0}$) estarem reproduint el contingut de manera que es conservaran totes les coordenades, amb el benentès que aquestes coordenades es referien al ***SCP*** vigent en definir el bloc i ara es referiran al ***SCP*** actual; i quan l'inserim amb valors explícits **O**, $\mathbf{E_x}$, $\mathbf{E_y}$ i $\boldsymbol{\alpha}$, el que fem és aplicar en aquesta reproducció passiva un escalat $\mathbf{E_x}$ i $\mathbf{E_y}$ (des de l'origen), un gir $\boldsymbol{\alpha}$ (al voltant de l'eix **Z**) i una translació segons el vector **O**, just en aquest ordre. Aclarit això, l'adaptació de la seqüència precedent a la lògica interna dels blocs d'AutoCAD seria:

Inserció de **BLOC**
1) Escalat uniforme $\mathbf{E_{1x} = E_{1y}}$ des de l'origen.
2) Gir $\boldsymbol{\alpha_1 + \alpha_3}$ al voltant de l'eix **Z**.
3) Translació des de l'origen al punt-base **O** del paral·lelogram.

4) Canvi de ***SCP***: desplaçament de l'origen a **O** i gir $\boldsymbol{\alpha_3}$ al voltant del nou eix **Z**.

Definició de **BLOC***
5) La figura resultant de 1, 2 i 3, queda referida al ***SCP*** implantat en 4.

Inserció de **BLOC***
6) Escalat $\mathbf{E_{2x} = 1}$ i $\mathbf{E_{2y} \neq 1}$ des del nou origen **O**.
7) No hi ha gir.
8) No hi ha translació.

9) Retorn al ***SCP*** anterior.

O bé, amb lleugers canvis a 4 i 5 però resultant un **BLOC*** idèntic, l'equivalent:

Inserció de **BLOC**
1) Escalat uniforme $\mathbf{E_{1x} = E_{1y}}$ des de l'origen.
2) Gir $\boldsymbol{\alpha_1 + \alpha_3}$ al voltant de l'eix **Z**.
3) Translació des de l'origen al punt-base **O** del paral·lelogram.

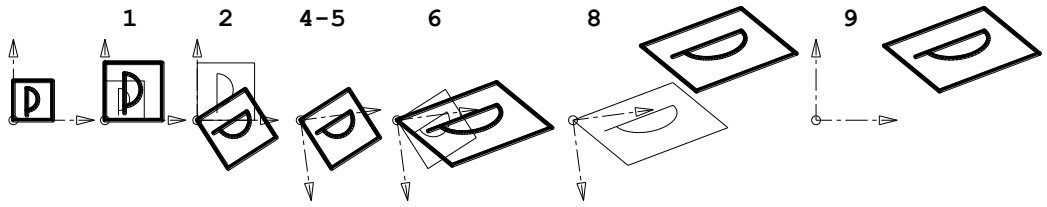
4) Canvi de ***SCP***: gir $\boldsymbol{\alpha_3}$ al voltant de l'eix **Z**.

Definició de **BLOC***
5) La figura resultant de 1, 2 i 3, amb el punt base situat a **O**, queda referida al ***SCP*** implantat en 4 però amb l'origen desplaçat fins aquest punt.

Inserció de **BLOC***
6) Escalat $\mathbf{E_{2x} = 1}$ i $\mathbf{E_{2y} \neq 1}$ des del nou origen **O**.
7) No hi ha gir.
8) No hi ha translació.

9) Retorn al ***SCP*** anterior.

Figura 8.7



La Figura 8.7 no il·lustra literalment cap de les dues seqüències presentades, sinó una 3ª versió, equivalent,

Inserció de **BLOC**
1) Escalat uniforme $\mathbf{E_{1x} = E_{1y}}$ des de l'origen.
2) Gir $\boldsymbol{\alpha_1 + \alpha_3}$ al voltant de l'eix **Z**.
3) No hi ha translació.

4)Canvi de ***SCP***: gir $\boldsymbol{\alpha_3}$ al voltant de l'eix **Z**.

Definició de **BLOC***
5) La figura resultant de 1, 2 i 3, queda referida al ***SCP*** implantat en 4.

Inserció de **BLOC***
6) Escalat $\mathbf{E_{2x} = 1}$ i $\mathbf{E_{2y} \neq 1}$ des de l'origen.
7) No hi ha gir.
8) Translació des de l'origen al punt-base **O** del paral·lelogram.

9) Retorn al ***SCP*** anterior.

per simplicitat de la pròpia figura i per poder comparar millor el procés amb el reflectit a la Figura 8.6 (si en **INSERT*** ens veiem obligats a inserir **BLOC** en **O**, és perquè l'angle $\alpha_1 + \alpha_3$ està definit per l'alineació **O–X'**).

De fet, si no ens haguéssim entestat a evitar que de cada execució de **C:INSERTOK** en resultés un **BLOC*** diferent, quan el seu contingut coincidís amb el d'altres pel que fa a l'angle α_1 girat per **BLOC**, el codi de **INSERT*** s'hauria pogut deixar així:

```
(defun INSERT* (/ X* X** Y** OX* OX** BLOC*)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**) K 0)
  (while (tblsearch "BLOCK" (setq K (1+ K) BLOC* (strcat BLOC "_" (itoa K)))))
  (command "CLAYER" "0"
           "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*
         ; "SCP" "DE" O
         ; "SCP" "Z" '(0 0) (mapcar '- X** O)
         ; "BLOQUE" BLOC* '(0 0) "LT" ""
         ; Les tres línies precedents corresponen a la 1ª versió de l'esquema.
         ; Les dues línies  següents  corresponen a la 2ª versió.
           "SCP" "Z" (setq K (trans O 1 0) O O) X**
           "BLOQUE" BLOC* (setq K (trans K 0 1)) "LT" ""
           "CLAYER" CAPA
           "INSERT" BLOC* K OX** (distance Y** Y) 0
           "SCP" "PR"))
```

Però des de bon començament havíem deixat clar que calia minimitzar la redundància del sistema definint uns genèrics **BLOC*** susceptibles de ser utilitzats en un màxim nombre d'aplicacions **C:INSERTOK**. Així que no hi havia més remei que complicar la funció **INSERT***, en la forma que l'hem mostrat abans, per donar cabuda a dos casos:
- Generació de **BLOC*** quan es presenta per primer cop la necessitat d'usar-ne un en què $\cos \alpha_1 = \dfrac{\mathbf{O–X''}}{\mathbf{O–X'}}$ tingui un valor diferent, i inserció del nounat per encabir en el paral·lelogram **X–O–Y** el quadrat $\mathbf{1 \times 1}$ associat a **BLOC**.
- Noves insercions d'aquest **BLOC*** cada vegada que comprovem que respon al valor $\cos \alpha_1 = \dfrac{\mathbf{O–X''}}{\mathbf{O–X'}}$ previst.

En el primer cas podríem haver aprofitat l'esquema precedent, però no tenia gaire sentit singularitzar la primera inserció quan estàvem obligats a abordar les que vinguessin a continuació, amb un **BLOC*** adient ja creat i des de sistemes diferents al ***SCP*** vigent en aquella primera ocasió: només caldrà retornar al ***SCP*** previ just després d'haver definit **BLOC***, i així la inserció inaugural podrà ser tractada amb la mateixa expressió muntada per a les posteriors. En l'esquema, això no suposarà únicament traslladar el punt 9 abans del 6 sinó reactivar els punts 7 i 8 (que ara anomenarem 8 i 9):

Inserció de **BLOC**
1) Escalat uniforme $\mathbf{E_{1x} = E_{1y}}$ des de l'origen.
2) Gir $\alpha_1 + \alpha_3$ al voltant de l'eix **Z**.
3) Translació des de l'origen al punt-base **O** del paral·lelogram.

4) Canvi de ***SCP***: gir α_3 al voltant de l'eix **Z**.

Definició de **BLOC***

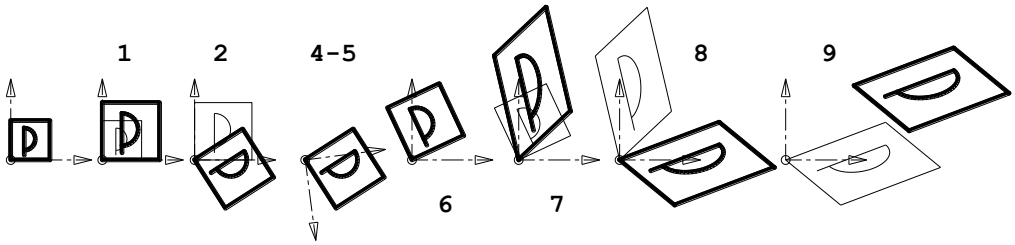
5) La figura resultant de 1, 2 i 3, amb el punt base situat a **O**, queda referida al ***SCP*** implantat en 4 però amb l'origen desplaçat fins aquest punt.

6) Retorn al ***SCP*** anterior.

Inserció de **BLOC***

7) Escalat $\mathbf{E_{2x} = 1}$ i $\mathbf{E_{2y} \neq 1}$ des de l'origen.
8) Gir $\boldsymbol{\alpha_3}$ al voltant de l'eix **Z** (en les noves insercions el gir serà $\boldsymbol{\alpha \neq \alpha_3}$).
9) Translació des de l'origen al punt-base **O** del paral·lelogram.

Figura 8.8



Per les raons adduïdes abans, la Figura 8.8 no mostra literalment la seqüència presentada, sinó l'equivalent

Inserció de **BLOC**

1) Escalat uniforme $\mathbf{E_{1x} = E_{1y}}$ des de l'origen.
2) Gir $\boldsymbol{\alpha_1 + \alpha_3}$ al voltant de l'eix **Z**.
3) No hi ha translació.

4)Canvi de ***SCP***: gir $\boldsymbol{\alpha_3}$ al voltant de l'eix **Z**.

Definició de **BLOC***

5) La figura resultant de 1, 2 i 3, queda referida al ***SCP*** implantat en 4.

6) Retorn al ***SCP*** anterior (s'hi ha dibuixat **BLOC*** referit a aquest ***SCP***).

Inserció de **BLOC***

7) Escalat $\mathbf{E_{2x} = 1}$ i $\mathbf{E_{2y} \neq 1}$ des de l'origen.
8) Gir $\boldsymbol{\alpha_3}$ al voltant de l'eix **Z** (en les noves insercions el gir serà $\boldsymbol{\alpha \neq \alpha_3}$).
9) Translació des de l'origen al punt-base **O** del paral·lelogram.

on el punt 6 de la figura pretén representar no només el retorn al ***SCP*** inicial (altrament, el quadrat no apareixeria girat) sinó el moment zero de la inserció (allò que abans anomenàvem repròducció passiva del bloc, prèvia a l'aplicació d'un escalat, un gir i una translació explícits) en aquest ***SCP***.

Dediquem-nos ara a aspectes no geomètrics de **C:INSERTOK** que cal no menystenir.

Si pensem que els components de **BLOC** poden pertànyer a la capa **"0"** o a qualsevol capa estàndard (anomenarem així les demés, per contraposició a la singularitat de la primera), i que les seves propietats color, tipus de línia i gruix de línia poden tenir valors específics o respondre a les conductes **"PORCAPA"** o **"PORBLOQUE"**, les característiques d'una inserció simple (pertinença a una capa i valor de les propietats esmentades, context que es tradueix en un determinat aspecte visual) obeiran a un ric ventall de combinacions, varietat no incompatible amb un rígid determinisme: si l'usuari domina les regles del joc pot preveure'n el resultat, sempre que conegui les condicions de partença (característiques originals dels components) i les de l'entorn de treball (característiques actuals). Ara bé, si allò que està en qüestió és el resultat d'inserir **BLOC*** i aquest és una inserció de **BLOC**, no n'hi haurà prou a conèixer les condicions de partença (prèvies a la definició de **BLOC**) i les de l'entorn (les vigents a l'hora d'inserir **BLOC*** per mitjà de **C:INSERTOK**): caldria conèixer també les vigents en executar **C:INSERTOK** just quan va donar lloc a la creació del genèric **BLOC***; només així, l'usuari amb sòlida formació podria seguir mentalment el procés i avançar una predicció fiable. Però això no és cap concurs per veure qui l'encerta o qui en sap més, d'AutoCAD, sinó una aplicació que l'usuari ha decidit d'utilitzar per arribar, en l'aspecte geomètric, fins allà on ***INSERT*** no podia dur-lo; però a banda de la deformació per cisallament, espera trobar els components de **BLOC** en les capes d'origen (tret que alguna d'elles fos la capa **"0"**) i veure'ls amb l'aspecte original (tret que el

valor original d'alguna propietat fos **"PORBLOQUE"**), com si l'operació s'hagués reduït a una inserció simple. I això només ho podrem garantir si, en la referida execució de **C:INSERTOK** (el primer cop que se les va veure amb un paral·lelogram d'acollida d'on resultava un valor $\frac{O-X''}{O-X'}$ diferent al dels genèrics existents), la inserció girada de **BLOC** que va permetre crear un nou **BLOC*** es va realitzar en les condicions ***CLAYER*** = **"0"** i ***CECOLOR*** = ***CELTYPE*** = ***CELWEIGHT*** = **"PORBLOQUE"**, perquè:

- Només amb ***CLAYER*** = **"0"**, cadascun dels components se situarà en la capa original, incloent-hi els situats a la capa **"0"** (se situen en la capa actual, que hem fet que també sigui la **"0"**).
- Només amb ***CECOLOR*** = ***CELTYPE*** = ***CELWEIGHT*** = **"PORBLOQUE"**, a cada un dels components totes tres propietats adoptaran el valor original, incloent-hi els que en tenien alguna en mode **"PORCAPA"** (tant si la capa és estàndard com si és **"0"** mantindrà el valor original **"PORCAPA"**: en el primer cas això remetrà al valor propi de la capa original, i en el segon, al valor de la capa actual, que és la **"0"** i per tant també l'original) o **"PORBLOQUE"** (adoptarà el valor actual, que hem fet que també sigui **"PORBLOQUE"**, i el component seguirà veient-se blanc o negre, segons el color del fons, i/o dibuixat amb línia contínua i/o amb el gruix per defecte establert per la variable ***LWDEFAULT***).

Com que, si es realitza una inserció de bloc en la capa **"A"** i la convertim en un bloc derivat que inserim en la capa **"B"**, el conjunt dels components del bloc primitiu deixa de visualitzar-se en pantalla tant si congelem **"A"** com si ho fem amb **"B"**, podríem sospitar que aquest tipus de manipulació posarà en evidència la veritable naturalesa de les insercions de **BLOC***: el conjunt de components de **BLOC** deixaria de veure's tant congelant la capa **"0"** com la capa actual en executar **C:INSERTOK**, i en això el comportament de les insercions no trivials realitzades amb ***INSERTOK*** diferiria de les realitzades amb ***INSERT***. Doncs no, fins i tot amb això ***INSERT*** i ***INSERTOK*** respondran d'igual manera: com que la inserció de **BLOC** que dóna lloc a **BLOC*** es fa a la capa **"0"**, en inserir **BLOC*** en una capa diferent, el component únic de **BLOC*** que és la inserció de **BLOC** passarà a residir en la segona capa; per tant, únicament si congelem aquesta capa, el conjunt dels components de **BLOC** desapareixerà. Així doncs, la funció **INSERT*** hauria de quedar en la forma

```
(defun INSERT* (/ CAPA COL TLIN GLIN X* X** Y** OX* OX** BLOC*)
  (setq CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0))))
        J 1 K (tblsearch "BLOCK" BLOC*))
  (if K
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 K))))) 0) -1 1)))
      (command "CLAYER" "0"
               "CECOLOR" "PORBLOQUE"
               "CELTYPE" "PORBLOQUE"
               "CELWEIGHT" -2 ; Valor equivalent a "PORBLOQUE", que no accepta.
               "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*
               "SCP" "Z" (setq K (trans O 1 0) O O) X**
               "BLOQUE" BLOC* (trans K 0 1) "LT" ""
               "SCP" "PR"
               "CELWEIGHT" GLIN
               "CELTYPE" TLIN
               "CECOLOR" COL
               "CLAYER" CAPA))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))
```

tot i que les quatre primeres assignacions les passarem a l'inici de **C:INSERTOK**, agrupant-les amb les **ECO** i **OSN**, per afavorir la claretat de lectura del codi.

En presentar la primera versió del codi, basada en l'aplicació estricta del mètode de Ritz, ja hi hem incorporat una primera comprovació de la idoneïtat dels ínputs, en aquell cas geomètrics, que s'ha mantingut: ens referim a la detecció de casos espuris (**X**, **O** i **Y** alineats o coincidents), que ens hem limitat a encarrilar cap a via morta sense molestar-nos a treure cap missatge, i a la qualificació de l'angle **X–O–Y**, per resoldre l'encaix amb una inserció simple de **BLOC** (recte) o, posats a fer-ho amb una doble inserció (agut o obtús), per escollir entre les dues variants de **BLOC*** lligades a un mateix cos α_1 de **BLOC** (en l'esmentada primera versió) o per orientar la construcció en el sentit adient (en l'actual). Quedava per comprovar que el nom **BLOC** respongués a algun objecte gràfic inserible: un bloc definit en el propi dibuix, que no hauria de dependre de referències externes (però que podria tenir-ne), o un dibuix localitzable (perquè **BLOC** inclou l'adreça de l'arxiu .DWG o perquè aquest es troba en alguna de les rutes de cerca prefixades), no enllaçat ni superposat com a referència externa. Ens hi posarem, però sense entretenir-nos amb la remota possibilitat que algú tingui la mala pensada d'utilitzar blocs "anònims" (codi_70 = **1**): blocs amb noms (valgui la paradoxa) del tipus ***D**<**n**> (que correspon a una cota associativa) o ***E**<**n**> (que correspon a la inserció girada d'un altre bloc o a un sòlid 3D, resultants de la descomposició d'insercions realitzades amb escalat no uniforme); noms del tipus ***X**<**n**> ja no n'hi ha (des d'AutoCAD v.14, les aplicacions de trama no tenen format de bloc i són objectes específics "**HATCH**").

La solució més simple seria, just després d'introduir el nom **BLOC**, fer

```
(setq ECO (getvar "CMDECHO"))
(setvar "CMDECHO" 0)
(command "INSERT" BLOC () "CMDECHO" ECO)
```

i deixar que fos l'Editor de Dibuix qui detectés hipotètiques anomalies: quan **BLOC** reunís les condicions enunciades no passaria res (en el cas d'un dibuix, a més, la cancel·lació de l'ordre no impediria que **BLOC** quedés integrat com a bloc propi, i això faria possible l'accés a la taula de símbols per esbrinar-ne la composició i característiques) i quan no fos així ja ens ho faria saber. El desavantatge és que l'avaluació només quedaria interrompuda quan fos **BLOC** = "": quan no hi hagués cap bloc ni cap dibuix **BLOC**, o quan sota aquest nom s'amagués una referència externa, sortiria el missatge adient però l'execució de **C:INSERTOK** proseguiria fins que un inevitable nou error la deturés. Per resoldre'l, forçant la interrupció en aquests supòsits, ho hauríem d'anar complicant, complicació que augmentaria en procurar que la variable ***CMDECHO*** no resultés afectada quan **BLOC** = "" i que la integració d'un dibuix com a bloc no provoqués més endavant una fallada, quan en el nom **BLOC** l'usuari hagués explicitat l'adreça de l'arxiu, a l'hora d'usar-lo literalment com a nom del bloc. Tot plegat ens conduiria al codi

```
(setq ECO (getvar "CMDECHO"))
(setvar "CMDECHO" 0)
(command "INSERT" (progn (if (= BLOC "") (setvar "CMDECHO" ECO)) BLOC) ()
         "CMDECHO" ECO)
(if (/= (substr (getvar "LASTPROMPT") 1 24) "Indique nombre de bloque") (exit))
(if (wcmatch BLOC "*:*,*/*,*\\*")
    (progn
      (setq B BLOC J (strlen B) K 1)
      (while (and (> J 0) (not (wcmatch (substr B J 1) ":,/,\\")))
             (setq BLOC (substr B J K) J (1- J) K (1+ K)))))
```

prou inflat com perquè ens plantegem si no seria millor buscar una alternativa en què el control de la situació passés de l'Editor de Dibuix a **C:INSERTOK**: potser hauríem d'inflar el codi encara un pèl més, però ho podríem compensar substituint el format dels tímids missatges estàndard d'error (que, en el cas dels llistats encapçalats per l'avís *No se encuentra el archivo en el camino de búsqueda: ...*, provoquen la commutació a finestra de text) per les més vistoses finestres **alert**. De fet, ens podem permetre complicar alguns aspectes perquè n'hem simplificat uns altres: ja us haureu adonat que, del missatge on se sol·licita el nom del bloc, hem prescindit de l'opció a veure la llista de noms dels blocs creats en el dibuix (el de l'ordre ***-INSERT*** és *Indique nombre de bloque o [?]:*, que en ***GINSERT*** havíem respectat pel seu caràcter de versió millorada de l'ordre bàsica, caràcter del qual no queda gens clar que participi ***INSERTOK***, concebuda per a d'altres usos). Encara que prenguem partit per l'alternativa esmentada, que presentem tot seguit, el lector sempre podrà substituir-la pel dispositiu precedent en les versions de **C:INSERTOK** que desfilaran per aquest capítol i els subsegüents.

Pel que fa a blocs definits en el propi dibuix i partint del supòsit expressat en el penúltim paràgraf (que a ningú se li acudirà invocar blocs dels mal anomenats "anònims", de codi_70 = **1**), no caldrà completar la condició d'accés

```
  (if (and (setq O (tblsearch "BLOCK" BLOC)) ...
```
amb expressions ajustades estrictament al valors **0** i **2** del codi 70, com
```
       ... (= (logand (cdr (assoc 70 O)) 125) 0)),
       ... (or (= (cdr (assoc 70 O)) 0) (= (cdr (assoc 70 O)) 2))) o
       ... (= (abs (1- (cdr (assoc 70 O)))) 1))
```
o de manera menys estricta (valors **0**, **1** i **2**), com
```
       ... (<= (cdr (assoc 70 O)) 2)),
```
sinó que ho deixarem encara més fàcil, fent
```
       ... (< (cdr (assoc 70 O)) 4))
```
Fixeu-vos que una cosa és voler progressar de forma sistemàtica, optant en el present capítol per un objectiu limitat com és la consecució de l'encaix 2D d'un bloc sense atributs, deixant per a més endavant la implementació de components d'aquesta mena per centrar-nos de moment en el problema geomètric, i una altra de ben diferent posar barreres als blocs amb atributs: per provar el bon funcionament d'aquesta primera versió ens limitarem a usar blocs que no en tinguin, però sense que les condicions d'accés siguin restrictives; per això considerem ambdues menes de blocs, amb atributs (codi_70 = **2**) o sense (codi_70 = **0**) i, acceptant que mai no cridarem blocs "anònims" (codi_70 = **1** + **2** o codi_70 = **1**), únicament ens previndrem contra els blocs temporals que donen suport a les referències externes mentre el dibuix és obert o contra els blocs que en depenen (codi_70 **≤** 4).

Pel que fa als dibuixos, descartat el supòsit que ja hagin estat inserits (**(setq O (tblsearch "BLOCK" BLOC))** i **(< (cdr (assoc 70 O)) 4)**), només resta escatir si són localitzables (**(not O)** i **(findfile (strcat BLOC ".dwg"))**), i amb aquest propòsit completarem la condició d'accés fent
```
  (if (or (and (setq O (tblsearch "BLOCK" BLOC))
               (< (cdr (assoc 70 O)) 4))
          (and (not O) (findfile (strcat BLOC ".dwg"))))
```
Si s'acompleix però **BLOC** encara no figura a la taula de blocs (**(not O)**, cosa que ens indica que era el nom d'un dibuix localitzable), convindria convertir-lo en bloc iniciant una inserció que cancel·laríem tan bon punt haguéssim introduït el nom (anàlogament a allò que fèiem en la primera versió), tant per homogeneitzar tots els casos resolutius com per deslliurar **BLOC** d'eventuals prefixos de ruta: tot plegat anirà a càrrec de la funció **DIBUIX**. Si no, el procés s'haurà de cloure (de forma "natural" o provocant la sortida amb **exit**), però abans caldria desplegar amb **alert** missatges adients: distingirem entre el cas de **BLOC** no localitzable com a bloc ni com a arxiu de dibuix (**(not O)** i **(not (findfile (strcat BLOC ".dwg")))**, a càrrec de **RUTES**, funció manllevada de l'aplicació **C:GINSERT** de l'últim capítol), i el cas de **BLOC** enllaçat o superposat com a referència externa o el de **BLOC** que en depèn d'una (**(tblsearch "BLOCK" BLOC))** i **(>= (cdr (assoc 70 O)) 4)**, a càrrec de la funció **REFX**). Les funcions auxiliars esmentades (incloent-hi **DEFECTE**, sobre la qual sobren explicacions) seran

```
(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (cdr (assoc 2 A)))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))
```

i, dins de **C:INSERTOK**, la condició d'accés comentada quedaria així

```
(defun C:INSERTOK (/ ...) ...
  (setq O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES))))))
  (if BLOC
      (progn
        (setq BLOC (strcase (if O BLOC (DIBUIX)))) ...)))
```

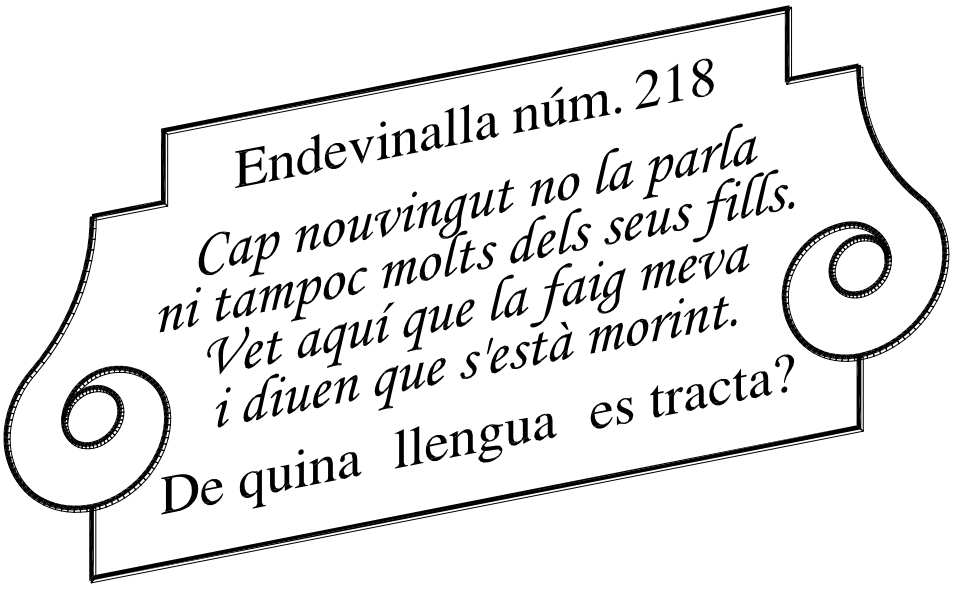
tot i que, per no entaforar tant la conclusió del codi, podríem deixar-ho així

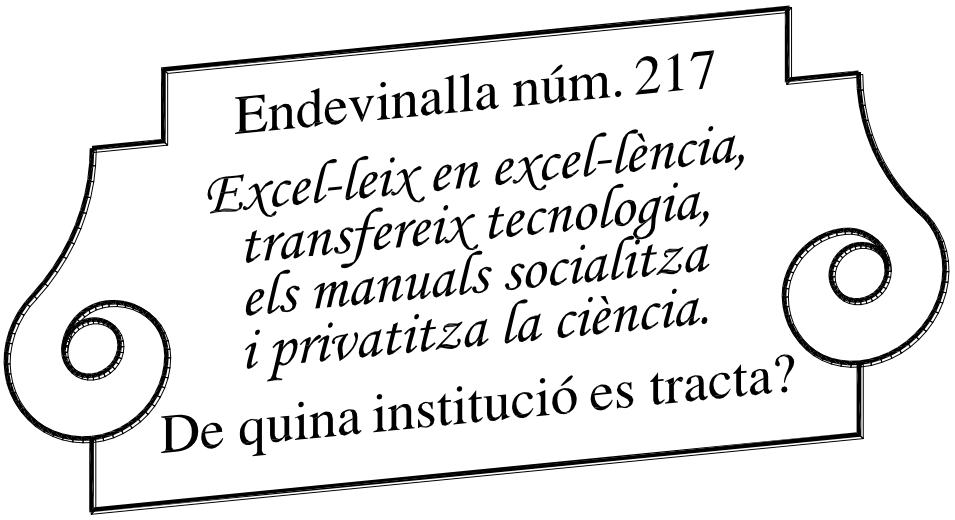
```
(defun C:INSERTOK (/ ...) ...
  (setq O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC
                 (strcase (if O BLOC (DIBUIX)))
                 (exit))) ...)
```

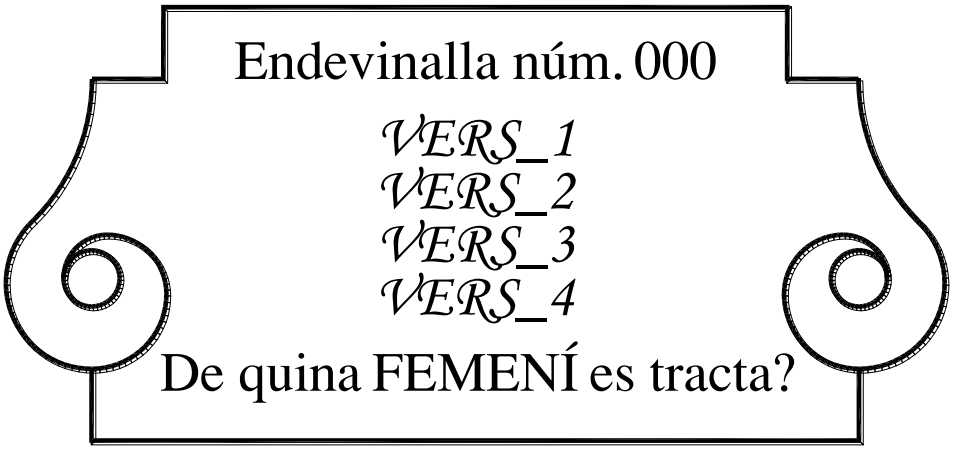
A la pràctica, l'accés a la funció **DIBUIX** quedarà postergat, produint-se des de **INSERT***. La raó és que, si el paral·lelogram d'acollida resulta ser un rectangle, no caldrà forçar la integració d'un dibuix alié en l'actual, convertint-lo en bloc propi i depurant-ne el nom **BLOC** per formar el nom **BLOC*** del genèric, perquè n'hi haurà prou amb una inserció simple de **BLOC** per assolir l'objectiu final. Aquest i altres detalls menors els trobareu a la versió completa del codi, que presentem tot seguit:

```
; VERSIÓ 1

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun INSERT* (/ X* X** Y** OX* OX** BLOC*)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
```

```
      X* (inters X (polar X W 1) O B ())
      X** (inters O (polar O (+ W PI/2) 1) X X* ())
      Y** (inters Y (polar Y W 1) O X** ())
      OX* (distance O X*)
      OX** (distance O X**)
      BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
      BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0))))
      J 1 K (tblsearch "BLOCK" BLOC*))
  (if K
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 K))))) 0) -1 1)))
      (command "CLAYER" "0"
               "CECOLOR" "PORBLOQUE"
               "CELTYPE" "PORBLOQUE"
               "CELWEIGHT" -2
               "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*
               "SCP" "Z" (setq K (trans O 1 0) O O) X**
               "BLOQUE" BLOC* (trans K 0 1) "LT" ""
               "SCP" "PR"
               "CELWEIGHT" GLIN
               "CELTYPE" TLIN
               "CECOLOR" COL
               "CLAYER" CAPA))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN BLOC O X Y OX OY XY A B
                     E LE W I J K)
  (setq Q0 0.001
        PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        O (getpoint "\nPunt d'insercio: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC  " <1>: "))
              W (angle O A)
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
  (if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
      (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
          (command "INSERT" BLOC O OX (* OY I) X)
          (INSERT*)))
  (setvar "INSNAME" BLOC)
  (setvar "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))
```


Endevinalla núm. 218
Cap nouvingut no la parla
ni tampoc molts dels seus fills.
Vet aquí que la faig meva
i diuen que s'està morint.
De quina llengua es tracta?


Endevinalla núm. 217
Excel-leix en excel-lència,
transfereix tecnologia,
els manuals socialitza
i privatitza la ciència.
De quina institució es tracta?


Endevinalla núm. 000
VERS_1
VERS_2
VERS_3
VERS_4
De quina FEMENÍ es tracta?

ENCAIX 2D DE BLOCS AMB ATRIBUTS

Adaptar **C:INSERTOK** a la presència en **BLOC** d'atributs normals (anomenarem així els no constants ni predefinits, siguin o no verificables i/o invisibles) no serà cap nimietat, no només per la complicació que introdueixen aquests components (caldrà formar en cada cas la llista **WWAA** de valors assignats) sinó perquè entrarà en joc un nou element discriminador: el conjunt de valors dels atributs de **BLOC** que, amb la deformació angular recollida en el paràmetre $\frac{O-X''}{O-X'}$, constituirà l'especificitat de cada genèric **BLOC***. Ja veurem en el decurs del present capítol com l'intent de superar això (deslligar els valors dels atributs de la caracterització de cada **BLOC***, tot desplaçant l'assignació des de la inserció de **BLOC** cap a la de **BLOC***), permet de passada anar prescindint de la llista **WWAA** i simplificar el panorama. Però més val que anem per pams i, pel que fa als pressupostos de partença (afegir la diferenciació textual a la geomètrica), en discutirem alguns aspectes abans de presentar el codi complet.

Hem parlat de complicació perquè, com ja havíem vist en el penúltim capítol en relació a **C:GINSERT**, la presència d'atributs normals en els blocs introdueix una incògnita que no ens permet despatxar les insercions amb l'expeditiva expressió **(command "INSERT" BLOC ...)**: ens referim al seu nombre, que es tradueix en un nombre igual d'arguments (eventualment incrementat en el d'atributs verificables) que caldrà afegir als que figuraven a la VERSIÓ 1 (si, pel que fa a les variables de sistema, ens trobem en les condicions habituals ***ATTREQ*** = **1**). Si allà ho havíem solucionat fent una llista **RR** d'expressions sense avaluar,

```
(repeat N (setq RR (cons '(if (> (getvar "CMDACTIVE") 0) PAUSE) RR)))
```

on **N** representava el màxim nombre d'atributs normals previsible, que fussionàvem amb la llista **'(command "INSERT" BLOC ...)**, forçant l'avaluació del conjunt com a expressió AutoLISP, més o menys d'aquesta manera,

```
(eval (append '(command "INSERT" BLOC "G" 0
                        (progn (setvar "CMDECHO" 1)
                               (prompt "Precise punto de inserción: ")
                               PAUSE))
              (repeat N (setq RR (cons '(if (> (getvar "CMDACTIVE") 0)
                                            PAUSE
                                            (if (= (getvar "CMDECHO") 1)
                                                (setvar "CMDECHO" 0)))
                                       RR)))))
```

aquí sembla que podríem explotar la mateixa fórmula, fent

```
(eval (append '(command "CLAYER" "0"
                        "CECOLOR" "PORBLOQUE"
                        "CELTYPE" "PORBLOQUE"
                        "CELWEIGHT" -2
                        "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I)
                                 (progn (setvar "CMDECHO" 1) X*))
              (repeat N (setq RR (cons '(if (> (getvar "CMDACTIVE") 0)
                                            (setq WA (getstring T)
                                                  WWAA (cons WA WWAA)
                                                  WA WA)
                                            (if (= (getvar "CMDECHO") 1)
                                                (setvar "CMDECHO" 0)))
                                       RR))) ...))
```

Tot i que haguem substituït la constant **PAUSE** per la funció **getstring**, per tal d'assignar el valor introduït a **WA** i poder formar la llista **WWAA** de valors d'atribut (atenció: havent usat **cons** per formar-la, caldrà donar-li la volta fent **(setq WWAA (reverse WWAA))**), necessària al seu torn per poder comparar aquests valors amb els d'altres genèrics **BLOC*** obtinguts inserint **BLOC** amb el mateix angle de gir (per veure si en podem aprofitar algun o n'hem de crear un altre), diverses raons ens duran a desestimar aquest procediment:

- La irrupció de l'eco del valor assignat a cada atribut, repetint en pantalla el que acabem d'escriure, després del canvi de línia provocat per l'imprescindible <Intro> i abans d'aparèixer el requeriment de valor de l'atribut següent (cosa que dificulta la seva neutralització, i que a la pràctica ens portaria a seguir en la situació **CMDECHO** = **0** i a reproduir els missatges de sol·licitud, copiant-

los de la definició de **BLOC** continguda a la taula de símbols).
- Quan ara mateix parlàvem del valor assignat a cada atribut, no ho fèiem amb propietat, perquè una cosa són els textos introduïts mitjançant **getstring** i transferits com arguments a **(command "INSERT" BLOC ...)**, i una altra els valors realment assignats. Només aquests últims haurien de figurar a la llista **WWAA**, i tanmateix hi ha dos casos en què ambdues sèries de dades no coincideixen:
  - Si l'atribut normal té valor per omissió, seria aquest el que hauria de figurar a la llista **WWAA**, i no pas el text nul introduït com a resposta.
  - Si té la característica *Verificar* activada, la seva presència es traduirà en dos arguments, dels quals només el segon representa el valor assignat que hauria de figurar a **WWAA**.

  Complicant l'algorisme es podrien detectar aquests supòsits i depurar **WWAA**, però encara quedaria una objecció de pes, que veiem tot seguit.
- Si després de la inserció descobríssim que ja hi havia un genèric **BLOC*** no només compatible geomètricament (**BLOC** gira un angle de cosinus $\frac{O-X''}{O-X'}$ igual a l'actual) sinó amb atributs d'idèntic valor, hauríem de fer marxa enrera i revocar-la.

Sembla doncs indiscutible que la demanda de valors per als atributs (valors que anirem emmagatzemant a **WWAA**, però no de manera mecànica sinó atenent el protocol exposat unes línies enrera) serà millor realitzar-la abans de la inserció de **BLOC** (si és que cal fer tal cosa per donar lloc a un nou **BLOC***), immediatament després que l'usuari hagi definit el paral·lelogram d'encaix. La funció **ATRIBS-1**, que se'n farà càrrec només quan **(= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2)**, és a dir, si el senyal associat al codi 70 en la definició de **BLOC** revela que hi ha atributs (condició aquesta necessària però no suficient, perquè podria ser que tots fossin constants i predefinits), només requerirà a l'usuari la introducció de valors quan la descripció **LE** de l'atribut acompleixi **(= (logand (cdr (assoc 70 LE)) 10) 0)**, és a dir, si el senyal associat al codi 70 revela que no és ni constant ni predefinit (expressat aquest valor en sistema binari, no pot tenir cap dígit **1** en comú amb **10 = 2 + 8**, ja que 2^1 s'associa a la característica *Constante* i 2^3 a *Predefinido*, condició aquesta que sí que és suficient), com ho faria la pròpia ordre **INSERT**. La funció **VATR** és l'ànima de **ATRIBS-1**, perquè formula les preguntes pertinents i en recull les respostes, substituint-les pels valors per omissió quan són nul·les. Si l'atribut és verificable s'accedirà una altra vegada a **VATR**: en aquesta segona volta, el resultat de la primera serà reintroduït com a valor per omissió; a més d'emmagatzemar-se el resultat en el lloc de la llista **WWAA** on li correspongui, se li afegirà per la cua una resposta nul·la de confirmació. Així doncs, **WWAA** quedarà constituïda pels textos que volem assignar als atributs normals (verificables o no), i a continuació per un seguit de textos "" (un per cada atribut verificable). Per tant, quan **WWAA** es fussioni amb **'(command "INSERT" BLOC ...)** i sigui avaluada, les respostes als atributs verificables no reproduiran necessàriament la seqüència viscuda en **ATRIBS-1**, sinó que en primera volta ja s'assignaran valors definitius (siguin introduïts aquests valors en 1ª o 2ª volta de **VATR**) i la segona volta sols farà que ratificar-los.

Aquesta llista ens permetrà rastrejar la taula de blocs, per veure si ja disposem del **BLOC*** adeqüat; si no fos així (i només en aquest cas), muntaríem el dispositiu

```
(eval (append '(command "CLAYER" "0"
                        "CECOLOR" "PORBLOQUE"
                        "CELTYPE" "PORBLOQUE"
                        "CELWEIGHT" -2
                        "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*)
              (if (= (getvar "ATTREQ") 1) WWAA)))
             '("SCP" "Z" (setq K (trans O 1 0) O O) X**
               "BLOQUE" BLOC* (trans K 0 1) "LT" ""
               "SCP" "PR"
               "CELWEIGHT" GLIN
               "CELTYPE" TLIN
               "CECOLOR" COL
               "CLAYER" CAPA)))
```

per tal de crear-lo (en la versió precedent del dispositiu no calia fer intervenir ***ATTREQ*** perquè, un cop determinada geomètricament la inserció, el valor d'aquesta variable quedaba reflectida en el de ***CMDACTIVE***). No cal dir que els noms d'aquests blocs genèrics, quan **BLOC** tingui atributs normals (és a dir, si de **ATRIBS-1** n'ha sortit una llista **WWAA** no nul·la), no només inclourà el sufix amb l'especificitat geomètrica (el cosinus de l'angle amb què s'hi insereix **BLOC**) sinó un segon sufix

en referència al valor dels atributs normals: la forma més econòmica de resoldre-ho és, a manera de subíndex, un enter **N** que vagi numerant correlativament cada nou genèric **BLOC*** en què la inserció de **BLOC** tingui el mateix angle de gir però una combinació inèdita de valors d'atributs normals. D'acord amb això, la sintaxi serà <nom_de_**BLOC**>_<decimals_cosinus_angle_gir>_<**N**èsima_combinació_d'atributs_normals>.

La comparació de la llista **WWAA** amb els valors dels atributs normals de **BLOC** en els genèrics **BLOC*** compatibles geomètricament, es desenvoluparà dintre de **INSERT*** (les insercions trivials, en què el paral·lelogram d'acollida es reveli rectangle, en tindran prou amb **BLOC**) i anirà a càrrec de la funció **ATRIBS-2**, que per dur-la a terme passarà a la llista **VVAA** el contingut dels atributs de cada **BLOC*** candidat. Només quatre observacions, a propòsit de **ATRIBS-2**:

- Atès que no és aconsellable utilitzar reiteradament **append** per formar llistes, per la seva lentitud en relació a **cons**, si un **BLOC** amb elevat nombre d'atributs normals justifiqués la substitució de **(append VVAA (list (cdr (assoc 1 LE))))** per **(cons (cdr (assoc 1 LE)) VVAA)**, **(mapcar '(lambda (VA WA) ...) VVAA WWAA)** s'hauría de transformar en **(mapcar '(lambda (VA WA) ...) (reverse VVAA) WWAA)**.
- Recordant que la llista d'arguments **WWAA** serà més llarga que les llistes **VVAA** de valors d'atributs normals de **BLOC**, si n'hi ha de verificables, hom podria pensar que aquest dispositiu de comparació **(mapcar '(lambda (VA WA) ...) VVAA WWAA)** no funcionarà mentre no s'apliqui a **WWAA** la llei de Procust i se li amputi l'excés que presenta sobre **VVAA** (la cua de valors nuls de confirmació). No cal, perquè la concurrència de llistes desiguals en **mapcar** no origina conflicte: el nombre d'iteracions coincidirà amb el d'elements de la llista més curta.
- A diferència dels atributs en estat de preassignació (objectes **"ATTDEF"**) que analitzàvem a **ATRIBS-1**, dels atributs amb valor assignat, que pertanyen a una inserció (objectes **"ATTRIB"**) i figuren com a informació annexa, la base de dades només en recull els predefinits i els normals (formen una seqüència rematada pel senyal **"SEQEND"**), no pas els constants. Així doncs, si només cal comparar els atributs normals, la funció **ATRIBS-2** sols haurà de filtrar els predefinits, de manera que la condició **(= (logand (cdr (assoc 70 LE)) 10) 0)** es pot substituir per la més laxa **(= (logand (cdr (assoc 70 LE)) 8) 0)**.
- Ara bé: la qüestió no és si hem de fer servir una o altra condició, sinó si en fem servir alguna. De fet, als atributs predefinits se'ls assigna d'entrada un mateix valor, però amb ***ATREDIT*** o ***-ATREDIT*** el podríem modificar, i tindria sentit plantejar la comparació de valors amb caràcter integral (en aquest supòsit, la llista **WWAA** hauria d'haver recollit també el valor dels atributs predefinits), però el perill d'arribar a tenir insercions diferents (pel que fa a atributs) d'un mateix genèric **BLOC*** és més especulatiu que real: si n'editem una, estarem redefinint **BLOC*** i automàticament (o forçant la regeneració del dibuix) el canvi es propagarà a totes les demés, com constatarem més endavant, en posar a prova la VERSIÓ 4.

Aclarit això, presentem el codi complet:

```
; VERSIÓ 2

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))
```

```
(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun ATRIBS-1 (/ ICVP N M VD V LLAA)
  (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
  (while E
         (if (and (= (cdr (assoc 0 (setq LE (entget E)))) "ATTDEF")
                  (= (logand (setq ICVP (cdr (assoc 70 LE))) 10) 0))
             (progn
               (if (and (not LLAA) ATREQ)
                   (prompt "\nIndique valores de atributo"))
               (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LE))
                     (cdr (assoc 3 LE)) (cdr (assoc 1 LE)))
               (setq LLAA (cons (list W N M V) LLAA))))
         (setq E (entnext E)))
  (setq W ())
  (foreach LA (reverse LLAA)
    (if (car LA)
        (progn
          (if (and (not W) ATREQ)
              (prompt "\nVerificar valores de atributo"))
          (VATR (cons "" W) (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq WWAA (append WWAA (list V))))
  (setq WWAA (append WWAA W)))

(defun ATRIBS-2 (/ VVAA)
  (setq A (strcat BLOC* "_")
        B A
        W (strlen A))
  (tblnext "BLOCK" T)
  (while (and (not BL*EX) (setq LE (tblnext "BLOCK")))
         (if (= (substr (setq K (cdr (assoc 2 LE))) 1 W) A)
             (progn
               (setq BL*EX LE
                     E (cdr (assoc -2 LE))
                     B K
                     VVAA ())
               (while (and (setq E (entnext E))
                           (= (cdr (assoc 0 (setq LE (entget E)))) "ATTRIB"))
                 (if (= (logand (cdr (assoc 70 LE)) 8) 0)
                     (setq VVAA (append VVAA (list (cdr (assoc 1 LE)))))))
               (mapcar '(lambda (VA WA) (if (/= VA WA) (setq BL*EX ())))
                       VVAA WWAA))))
  (setq BLOC* (if BL*EX B (strcat A (itoa (1+ (atoi (substr B (1+ W)))))))))

(defun INSERT* (/ X* X** Y** OX* OX** BLOC* BL*EX)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))) J 1)
  (if WWAA
      (ATRIBS-2)
      (setq BL*EX (tblsearch "BLOCK" BLOC*)))
```

```
  (if BL*EX
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 BL*EX))))) 0)
                       -1 1)))
      (eval (append '(command "CLAYER" "0"
                              "CECOLOR" "PORBLOQUE"
                              "CELTYPE" "PORBLOQUE"
                              "CELWEIGHT" -2
                              "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*)
                    (if ATREQ WWAA)
                   '("SCP" "Z" (setq K (trans O 1 0) O O) X**
                     "BLOQUE" BLOC* (trans K 0 1) "LT" ""
                     "SCP" "PR"
                     "CELWEIGHT" GLIN
                     "CELTYPE" TLIN
                     "CECOLOR" COL
                     "CLAYER" CAPA))))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC WWAA O X
                     Y OX OY XY A B E LE W I J K)
  (setq Q0 0.001
        PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        O (getpoint "\nPrecise punto de inserción: "))
  (foreach P '("X" "Y")
       (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                   " desde el punto de inserción: "))
             I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                  BLOC " <1>: "))
             W (angle O A)
             B (/ (distance O A) (if I I 1)))
       (set (read P) (polar O W B))
       (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) (ATRIBS-1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
  (setvar "ATTDIA" 0)
  (if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
      (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
          (eval (append '(command "INSERT" BLOC O OX (* OY I) X)
                        (if ATREQ WWAA)))
          (INSERT*)))
  (setvar "INSNAME" BLOC)
  (setvar "ATTDIA" ATDIA)
  (setvar "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))
```

De tota manera, si mirem el codi amb atenció, ens adonarem que hi ha lloc per a una simplificació que n'arrossegarà d'altres: si **BLOC** té atributs normals i és ***ATTREQ* = 1**, quan el paral·lelogram d'acollida sigui oblic i haguem de recórrer a un derivat genèric **BLOC***, serà inevitable construir-lo inserint **BLOC** amb la intervenció de la llista **WWAA** de valors d'atribut i, per tant, caldrà recórrer a l'artifici de transformar **command** en una llista sense avaluar (el resultat de refondre **WWAA**, de longitud indeterminada, amb els fragments de l'expressió que disposen d'un nombre determinat d'arguments); però quan el paral·lelogram d'encaix sigui rectangle, cas que es reconduïrà cap a una inserció ordinària de **BLOC**, la intervenció de **WWAA** serà perfectament prescindible. La causa és que, si falten arguments quan executem ordres AutoCAD mitjançant una funció **command** ubicada en una expressió AutoLISP més àmplia, l'Editor de Dibuix els buscarà en la següent funció **command** que s'avalui dintre de la mateixa expressió, i si no n'hi hagués cap més, un cop executada aquesta, assumirà directament el control i quedarà a l'espera de les dades pendents, que l'usuari haurà de subministrar en temps real. Això vol dir que, si en el primer cas no aportéssim la llista **WWAA** amb els últims arguments per a ***-INSERT***, AutoCAD aniria prenent els que vinguessin a continuació (els textos **"SCP"** i **"Z"**, els que reprodueixen els valors numèrics **K** i **X****, el text **"BLOQUE"**, el nom de **BLOC***, etc.), distorsionant amb tot això l'execució prevista. Però en el segon, si no li subministrem **WWAA**, es limitarà a avaluar les funcions terminals **setvar** i tornarà reclamar la introducció de valors per als atributs (per cert: **(setvar "INSNAME" BLOC)** no hi fa res, perquè no serà fins a rematar l'ordre ***-INSERT*** en curs que AutoCAD assignarà a la variable el nom del bloc utilitzat, que sortosament és **BLOC**). En aquest darrer cas, prescindirem de **ATRIBS-1**: per no haver de contestar dos cops les mateixes preguntes i perquè tampoc no necessitem **WWAA** en una comprovació **ATRIBS-2** que ja no cal fer, en treballar directament amb **BLOC**. Totes les possibilitats suggerides (supressió de l'argument **WWAA** en el cas de la inserció simple de **BLOC**, amb avaluació directa de **command** i exclusió de **ATRIBS-1**, funció que queda confinada a l'àmbit de **INSERT***), es materialitzen en el nou codi:

```
; VERSIÓ 3

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun ATRIBS-1 (/ ICVP N M VD V LLAA)
  (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
  (while E
         (if (and (= (cdr (assoc 0 (setq LE (entget E)))) "ATTDEF")
                  (= (logand (setq ICVP (cdr (assoc 70 LE))) 10) 0))
```

```
            (progn
              (if (and (not LLAA) ATREQ)
                  (prompt "\nIndique valores de atributo"))
              (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LE))
                    (cdr (assoc 3 LE)) (cdr (assoc 1 LE)))
              (setq LLAA (cons (list W N M V) LLAA))))
          (setq E (entnext E)))
  (setq W ())
  (foreach LA (reverse LLAA)
    (if (car LA)
        (progn
          (if (and (not W) ATREQ)
              (prompt "\nVerificar valores de atributo"))
          (VATR (cons "" W) (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq WWAA (append WWAA (list V))))
  (setq WWAA (append WWAA W)))

(defun ATRIBS-2 (/ VVAA)
  (setq A (strcat BLOC* "_")
        B A
        W (strlen A))
  (tblnext "BLOCK" T)
  (while (and (not BL*EX) (setq LE (tblnext "BLOCK")))
         (if (= (substr (setq K (cdr (assoc 2 LE))) 1 W) A)
             (progn
               (setq BL*EX LE
                     E (cdr (assoc -2 LE))
                     B K
                     VVAA ())
               (while (and (setq E (entnext E))
                           (= (cdr (assoc 0 (setq LE (entget E)))) "ATTRIB"))
                 (if (= (logand (cdr (assoc 70 LE)) 8) 0)
                     (setq VVAA (append VVAA (list (cdr (assoc 1 LE)))))))
               (mapcar '(lambda (VA WA) (if (/= VA WA) (setq BL*EX ())))
                       VVAA WWAA))))
  (setq BLOC* (if BL*EX B (strcat A (itoa (1+ (atoi (substr B (1+ W)))))))))

(defun INSERT* (/ X* X** Y** OX* OX** BLOC* BL*EX E LE)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))) J 1)
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) (ATRIBS-1))
  (if WWAA
      (ATRIBS-2)
      (setq BL*EX (tblsearch "BLOCK" BLOC*)))
  (if BL*EX
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 BL*EX))))) 0)
                       -1 1)))
      (eval (append '(command "CLAYER" "0"
                              "CECOLOR" "PORBLOQUE"
                              "CELTYPE" "PORBLOQUE"
                              "CELWEIGHT" -2
                              "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*)
                    (if ATREQ WWAA)
                    '("SCP" "Z" (setq K (trans O 1 0) O O) X**
                      "BLOQUE" BLOC* (trans K 0 1) "LT" ""
                      "SCP" "PR"
                      "CELWEIGHT" GLIN
                      "CELTYPE" TLIN
```

```
                      "CECOLOR" COL
                      "CLAYER" CAPA))))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC WWAA O X
                     Y OX OY XY A B W I J K)
  (setq Q0 0.001
        PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        O (getpoint "\nPrecise punto de inserción: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC " <1>: "))
              W (angle O A)
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
  (setvar "ATTDIA" 0)
  (if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
      (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
          (progn
            (command "INSERT" BLOC O OX (* OY I) X)
            (prompt (if (> (getvar "CMDACTIVE") 0)
                        "\nIndique valores de atributo\n"
                        "\n")))
          (INSERT*)))
  (setvar "INSNAME" BLOC)
  (setvar "ATTDIA" ATDIA)
  (setvar "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))
```

Tot funciona, tret del missatge *Verificar valores de atributos*, que apareix quan hi ha atributs verificables (per no estar sotmès a la variable de sistema ***CMDECHO***) i del qual ja ens en ocuparem més endavant, quan ataquem les qüestions formals). Adoneu-vos, però, que aquesta caracterització de **BLOC*** i del seu nom ens durà, a la pràctica, a veure reflectit allò de "tants caps, tants barrets" en l'estructura del blocs creada per **C:INSERTOK:** tret que totes les insercions de **BLOC** haguessin de tenir els mateixos valors d'atribut (cas en què n'hi hauria prou amb un **BLOC*** genèric per a cada angle de cosinus $\frac{O\text{–}X''}{O\text{–}X'}$, però en què seria més lògic haver usat textos i no atributs), si per a cada angle d'inserció hi ha d'haver tants genèrics **BLOC*** com combinacions de valors d'atribut, gairebé en sortirà un per inserció,

atès que la probabilitat d'insercions amb atributs idèntics serà baixa (tant més baixa quant major el nombre d'atributs). Només hi hauria una forma d'evitar-ho: que l'assignació de valor (almenys, l'assignació definitiva) es realitzés no a l'hora d'inserir **BLOC** per constituir el genèric **BLOC***, sinó en inserir aquest darrer: així, en la mesura que **BLOC*** fos previ a uns valors textuals diferents en cada cas (com ho és als factors d'escala i al gir que fan possible el seu encaix en un paral·lelogram determinat) seria un bloc realment genèric, d'utilització regida per criteris exclusivament geomètrics.

En aquesta línia, provarem d'assignar valors als atributs quan inserim **BLOC** per constituir cada **BLOC*** genèric, tal i com ja havíem fet, però només uns valors provisionals (assignant indiscriminadament el valor ".", per exemple), procedint a inserir **BLOC*** sense ocupar-nos d'aquests atributs (perquè són part de **BLOC**, una inserció del qual és el component solitari de **BLOC***) però editant-los tot seguit amb l'ordre ***-ATREDIT*** (recordeu que via **command** podeu recórrer a ***ATREDIT***, amb idèntics resultats). En principi la idea no és forassenyada: contra qualsevol expectativa, ***-ATREDIT*** no limita la seva acció als atributs situats en un primer nivell (la inserció més externa) sinó que l'estén a nivells més profunds. Per ser precisos, hauríem d'aclarir que aquest comportament sols es dóna quan contestem afirmativament la pregunta *¿Editar atributos uno a uno?* o bé quan, després de donar-li una resposta negativa, contestem afirmativament la qüestió *¿Editar sólo atributos visibles en pantalla?*; únicament l'edició global d'atributs (ambdues respostes negatives) es queda en el nivell més superficial. Una altra cosa són les implicacions que té aquest sorprenent poder de penetració, que dissortadament no trigarem a descobrir. De moment, posarem fil a l'agulla per adaptar la versió precedent al nou plantejament. Com a diferències bàsiques assenyalem:

- La supressió de la funció **ATRIBS-2**, ja que ara **BLOC*** és independent dels valors dels atributs i el seu nom no ha de reflectir les diferències en aquest aspecte.
- La incorporació a la funció **ATRIBS-1** (que ara anomenem **VAL-ATRIBS**) de noves llistes: **NNAA**, amb els identificadors dels atributs (que ***-ATREDIT*** necessitarà, per acotar cada selecció gràfica a un tipus d'atribut, assumit que a **BLOC** només n'hi pot haver un de cada), i **VVAA**, amb els valors provisionals que adjudicarem quan inserim **BLOC** per constituir **BLOC*** (no poden ser textos nuls, "", perquè la posterior selecció gràfica a ***-ATREDIT*** no els detectaria).
- Com ja havíem assenyalat en el penúltim capítol, en relació a **C:RATREDIT**, quan l'ordre ***-ATREDIT*** és executada des de **command** sols admet un atribut per inserció: aquí, on només en tenim una per editar (encara el bloc portador de l'atribut en primera instància sigui **BLOC**, ens haurem de referir a les insercions de **BLOC***, i per identificar-ne l'última només disposarem del valor actual ".", assumint que l'usuari mai no haurà assignat aquest valor), no tindrem altre remei que iterar tants cops com atributs normals tingui **BLOC**, ubicant l'ordre en el dispositiu **(mapcar '(lambda (NA VA) ...) NNAA WWAA)** per poder editar-los tots.
- Encara que la llista **WWAA** incorpori una cua de valors "" per a la confirmació d'atributs verificables, l'esmentat dispositiu funcionarà correctament, com ja assenyalàvem en la VERSIÓ 2 a propòsit de **VVAA** i **WWAA** (funció **ATRIBS-2**).
- Les insercions que comporten simetria (**I < 0**), són percebudes com si s'haguessin realitzat sota un ***SCP*** amb l'eix ***Z*** orientat a l'inrevés, raó per la qual ***-ATREDIT*** no podria editar-ne els atributs (*atributos no paralelos al SCP*) sense invertir-lo abans, fent **(command "SCP" "3" "" "" "0,-1")** o **(command "SCP" "X" 180)**.

```
; VERSIÓ 4

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))
```

```
(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS (/ E LE ICVP M N VD V LLAA)
  (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
  (while E
         (if (and (= (cdr (assoc 0 (setq LE (entget E)))) "ATTDEF")
                  (= (logand (setq ICVP (cdr (assoc 70 LE))) 10) 0))
             (progn
               (if (and (not LLAA) ATREQ)
                   (prompt "\nIndique valores de atributo"))
               (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LE))
                     (cdr (assoc 3 LE)) (cdr (assoc 1 LE)))
               (setq LLAA (cons (list W N M V) LLAA))))
         (setq E (entnext E)))
  (setq W ())
  (foreach LA (reverse LLAA)
    (if (car LA)
        (progn
          (if (and (not W) ATREQ) (prompt "\nVerificar valores de atributo"))
          (VATR (cons "" W) (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq NNAA (append NNAA (list (cadr LA)))
          VVAA (cons "." VVAA)
          WWAA (append WWAA (list V))))
  (setq VVAA (append VVAA W)
        WWAA (append WWAA W)))

(defun INSERT* (/ X* X** Y** OX* OX** BLOC* BL*EX C1 C2)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) (VAL-ATRIBS))
  (if (setq BL*EX (tblsearch "BLOCK" BLOC*))
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 BL*EX))))) 0)
                       -1 1)))
      (progn
        (setq J 1 K (/ OX* OX**))
        (eval (append '(command "CLAYER" "0"
                                "CECOLOR" "PORBLOQUE"
                                "CELTYPE" "PORBLOQUE"
                                "CELWEIGHT" -2
                                "ATTREQ" 1
                                "INSERT" BLOC O K (* K I) X*) VVAA
                      '("SCP" "Z" (setq K (trans O 1 0) O O) X**
                        "BLOQUE" BLOC* (trans K 0 1) (entlast) ""
                        "SCP" "PR"
                        "ATTREQ" (if ATREQ 1 0)
```

```
                        "CELWEIGHT" GLIN
                        "CELTYPE" TLIN
                        "CECOLOR" COL
                        "CLAYER" CAPA)))))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**)
  (if WWAA
      (progn
        (if (< I 0) (command "SCP" "X" "180"))
        (command "ZOOM" "T")
        (setq C1 (getvar "VSMIN")
              C2 (getvar "VSMAX"))
        (mapcar
          '(lambda (NA WA)
             (command "ATREDIT" "" BLOC* NA "." "C" C1 C2 "V" "" WA ""))
          NNAA WWAA)
        (command "ZOOM" "P")
        (if (< I 0) (command "SCP" "PR")))))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC NNAA VVAA
                     WWAA O X Y OX OY XY A B W I J K)
  (setq Q0 0.001
        PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        O (getpoint "\nPrecise punto de inserción: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC " <1>: "))
              W (angle O A)
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
  (setvar "ATTDIA" 0)
  (if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
      (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
          (progn
            (command "INSERT" BLOC O OX (* OY I) X)
            (prompt (if (> (getvar "CMDACTIVE") 0)
                        "\nIndique valores de atributo\n" "\n")))
          (INSERT*)))
  (setvar "INSNAME" BLOC)
  (setvar "ATTDIA" ATDIA)
  (setvar "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))
```

Mentre executem aquesta versió de **C:INSERTOK** per crear blocs genèrics **BLOC*** amb un determinat **BLOC** no tindrem cap contratemps (tret del missatge *1 encontrado(s)* que es deixa veure a cada iteració de ***-ATREDIT***), però quan el paral·lelogram on hagi de quedar encaixat el quadrat de referència associat a **BLOC** no comporti crear cap nou **BLOC***, perquè ja tinguem el que ens calia, el fiasco serà complet: la segona inserció mantindrà els valors d'atributs assignats en la primera i l'execució s'interromprà, treient els missatges *0 encontrado(s) *No válido* ; error: Función cancelada*. Si, per esbrinar què ha passat, se'ns acut aplicar ***-ATREDIT*** a aquestes dues insercions observarem un seguit de despropòsits però el resultat final anirà en la mateixa línia: les modificacions realitzades en una es propagaran a l'altra (tot i que caldrà forçar una regeneració del dibuix per copsar el seu abast). I si inserim "manualmemt" el bloc rebel (volem dir inserir amb ***INSERT*** el bloc genèric **BLOC***) tindrem una sorpresa, perquè no ens trobarem el que esperàvem: veurem **BLOC**, escalat uniformement (si adoptem els valors per defecte) i girat un cert angle, però els atributs reproduiran un altre cop els valors de les insercions oblíqües. Què passa? Molt senzill: quan ***-ATREDIT*** actua sobre els atributs d'una inserció, està actuant sobre totes les demés (passades i futures), perquè en realitat ho fa sobre una mateixa inserció de **BLOC** (la que forma part de la definició de **BLOC***); en definitiva, el que fem mitjançant ***-ATREDIT*** és redefinir **BLOC***, i això explica que després d'una primera inserció ja no quedi cap atribut de valor ".".

Si el problema era la profunditat d'allotjament dels atributs, podríem pensar a explosionar totes i cadascuna de les insercions de **BLOC***, immediatament després d'obtenir-les, i així n'alliberaríem el contingut, que pujaria un nivell: una inserció girada de **BLOC**, deformada per efecte de l'escalat no uniforme, amb valors d'atribut arbitraris que podríem editar amb ***-ATREDIT*** perquè ara estaríem actuant sobre una inserció de **BLOC** deslligada de la definició de **BLOC***, raó per la qual els canvis no provocarien cap redefinició (ni de **BLOC*** ni de **BLOC**). Què tindríem?: una inserció de **BLOC** amb els valors d'atribut desitjats. Doncs ¿per què no rematem sempre amb ***DESCOMP*** la inserció de **BLOC***, fins i tot quan **BLOC** no tingui atributs, desempallegant-nos així d'insercions que només necessitàvem com a cavall de Troia, si això ens ha de permetre quedar-nos amb insercions simples de **BLOC**? De fet, n'hi hauria prou a modificar la funció **INSERT*** en el codi subratllat:

```
; fragment de VERSIÓ 5
; ..........................................................
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**
           "DESCOMP" (entlast))
  (if WWAA
      (progn
        (command "ZOOM" "T")
        (setq C1 (getvar "VSMIN")
              C2 (getvar "VSMAX"))
        (mapcar
          '(lambda (NA WA)
             (command "ATREDIT" "" BLOC NA "." "C" C1 C2 "V" "" WA ""))
          NNAA WWAA)
        (command "ZOOM" "P"))))
```

Però no cal que us molesteu: tot i que, des de la versió 13 d'AutoCAD, ***DESCOMP*** ja és aplicable a insercions no escalades uniformement ($\mathbf{E_x \neq E_y}$), encara es mantenen irreductibles els blocs constituïts per insercions amb gir d'altres blocs. Aquesta limitació ja no és cosa contingent, d'una versió o una altra, sinó ontològica: el quadrat ortogonal de referència d'un bloc mai no podrà esdevenir paral·lelogram oblic per simple inserció. En el nostre cas (**BLOC*** format per una inserció de **BLOC** en solitari), la descomposició serà senzillament impossible. Quan el bloc extern constés d'altres objectes, a més d'insercions, ***DESCOMP*** l'arribaria a rebentar però la dura realitat també s'acabaria imposant, tot i que sota una altra aparença: si afegim a **BLOC*** qualsevol altre objecte (un punt, per exemple) i explosionem les seves insercions, les de **BLOC** canviaran d'identitat; ja no faran referència a un mateix bloc, sinó que cada inserció es referirà a un bloc diferent (noms del tipus ***E**<**n**>, ***E**<**n+1**>, ***E**<**n+2**> ...). Encara que els atributs d'aquests blocs anònims es poden editar, no sembla recomanable seguir aquest camí, complicant la definició de **BLOC*** només per arribar a consolidar allò de "tants caps, tants barrets" que era justament el que volíem evitar.

Més encertat seria explosionar **BLOC** després de la inserció prèvia a la constitució de cada bloc genèric **BLOC***, perquè **BLOC** haurà girat però amb un escalat uniforme ($\mathbf{E_x = |E_y|}$). Només caldria simplificar la funció **VAL-ATRIBS** (suprimir la llista

NNAA d'identificadors d'atributs normals, que només necessitàvem per acotar la selecció gràfica en ***-ATREDIT***) i modificar **INSERT*** en el codi subratllat:

```
; fragment de VERSIÓ 6
(defun INSERT* (/ X* X** Y** OX* OX** BLOC* BL*EX E LE)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0))) (if (< I 0) "-" "+"))
        J (1- (strlen BLOC*)))
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) (VAL-ATRIBS))
  (tblnext "BLOCK" T)
  (while (and (setq BL*EX (tblnext "BLOCK"))
              (/= (substr (cdr (assoc 2 BL*EX)) 1 J) (substr BLOC* 1 J))))
  (if BL*EX
      (setq BLOC* (cdr (assoc 2 BL*EX))
            J (* I (if (= (substr BLOC* (1+ J)) "-")
                       -1 1)))
      (progn
        (setq J 1 K (/ OX* OX**))
        (eval (append '(command "CLAYER" "0"
                                "CECOLOR" "PORBLOQUE"
                                "CELTYPE" "PORBLOQUE"
                                "CELWEIGHT" -2
                                "ATTREQ" 1
                                "INSERT" BLOC O K (* K I) X*) VVAA
                      '("DESCOMP" (entlast)
                        "SCP" "Z" (setq K (trans O 1 0) O O) X**
                        "BLOQUE" BLOC* (trans K 0 1) "P" ""
                        "SCP" "PR"
                        "ATTREQ" (if ATREQ 1 0)
                        "CELWEIGHT" GLIN
                        "CELTYPE" TLIN
                        "CECOLOR" COL
                        "CLAYER" CAPA)))))
  (eval (append '(command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**)
                (if ATREQ WWAA))))
```

Veiem que ha desaparegut l'epílog **(if WWAA ...)**, portador d'una ordre ***-ATREDIT*** que ja no farem servir, però d'altres coses es compliquen: com que la inserció de **BLOC** en què es fonamenta el genèric **BLOC*** s'ha explosionat i en les futures insercions d'aquest genèric ja no hi haurà forma de conèixer el signe del factor d'escala $\mathbf{E_y}$ utilitzat, incorporarem al nom de **BLOC*** un últim caràcter (**"+"** o **"-"**) indicatiu d'aquesta circumstància; també ens hem trobat que la interposició de ***DESCOMP*** entre ***-INSERT*** **BLOC** i ***-BLOQUE*** **BLOC*** obligava a embolcallar ***-INSERT*** **BLOC*** en el dispositiu **(eval (append ...))**, perquè aquesta era l'ocasió d'assignar als atributs els seus valors definitius **WWAA**. Ara bé, ¿segur que encara necessitem la llista **WWAA**, la de valors provisionals **VVAA** i la mare que les va parir, **VAL-ATRIBS**? Evidentment, no:

- Ja podem prescindir de la llista de valors d'atribut **WWAA** per inserir **BLOC***, com en prescindíem quan **O—X** i **O—Y** formaven angle recte i inseríem directament **BLOC**, des de la VERSIÓ 3. La raó és idèntica: sent l'última funció **command** avaluada per **C:INSERTOK** (en haver desaparegut l'epílog esmentat més amunt), si AutoCAD no hi troba tots els arguments que necessita per completar l'execució de ***-INSERT***, s'esperarà a la sortida de **C:INSERTOK** i requerirà l'usuari perquè li aporti les dades pendents. Això ens permet recuperar l'avaluació directa de **command** (sense el dispositiu **(eval (append ...))**) però ens obligarà a fer quelcom per assegurar que, la primera vegada que tornem a utilitzar **C:INSERTOK**, el nom del bloc que se'ns proposi per defecte sigui un altre cop **BLOC**: quan **BLOC** incorpora atributs i ***ATTREQ*** **= 1**, l'expressió terminal **(setvar "INSNAME" BLOC)** no pot acomplir la seva missió perquè ***-INSERT*** encara no s'haurà completat; si el paral·lelogram d'acollida és un rectangle (inserció de **BLOC**) això no importarà massa, però si no ho és (inserció de **BLOC***) el valor que finalment prendrà ***INSNAME*** serà el nom

de **BLOC***. De seguida presentarem el dispositiu corrector, però de moment seguim amb allò que dúiem entre mans.

- A diferència de les versions 4 i 5, en què l'assignació provisional de valor als atributs normals havia d'assegurar textos no nuls, per fer possible la posterior selecció gràfica en ***-ATREDIT***, ara tant se val quins valors adoptem quan inserim **BLOC** (en lloc de formar la llista **VVAA** amb valors "."), ho podríem haver fet amb valors ""), perquè el pas següent consistirà a explosionar aquesta inserció i tots els atributs tornaran a l'estat original de preassignació. A més, ¿per què molestar-nos a crear **VVAA**, si podem executar ***-INSERT*** sense assignació explícita de valors d'atribut, a condició de desactivar la variable de sistema ***ATTREQ***?: AutoCAD assignarà a cada atribut el seu valor per defecte (el previst en definir l'atribut o "", si no se n'havia previst cap), que com a valor provisional és tan bo com qualsevol altre; i sobretot, igual que en la posterior inserció de **BLOC***, també en la de **BLOC** podrem avaluar directament **command.**
- Conseqüència de les dues constatacions precedents, no tenint **VAL-ATRIBS** cap més missió que formar les llistes **VVAA** i **WWAA**, ens desempallegarem d'aquesta funció.

Pel que fa a la inoperància de l'expressió **(setvar "INSNAME" BLOC)**, que eliminarem de les assignacions finals, podríem recórrer a reactius tipus **vlr-LISP-reactor** i **vlr-COMMAND-reactor**, en combinació amb les variables ***INSNAME*** i ***BLOC*** (globals en la sessió de dibuix), per detectar quan una ordre AutoCAD iniciada dintre d'una expressió AutoLISP no s'acaba d'executar fins que la segona ja ha estat avaluada, i restablir **BLOC** com a nom de bloc per defecte, però el risc d'interferències amb altres aplicacions aconsella un mínim de prudència: ens limitarem a presentar el dispositiu, incorporant-lo al codi com a comentari (l'encapçalament ;; identifica les línies implicades), però n'usarem un de més simple, restringit a **C:INSERTOK** (de manera que, si després de la nostra aplicació executem ***INSERT***, aquesta ordre oferirà **BLOC*** i no **BLOC** com a nom de bloc per defecte). El que posarem en el lloc de **(setvar "INSNAME" BLOC)** és el canvi de línia (precedit per l'anunci d'atributs, si s'escau) que de la VERSIÓ 3 ençà únicament acompanyava les aplicacions trivials de **C:INSERTOK** (insercions simples) i que ara haurem de generalitzar: l'expressió **(prompt (if (> (getvar "CMDACTIVE") 0) "\nIndique valores de atributo\n" "\n"))**.

```
; VERSIÓ 7

;;  (vl-load-com)
;;  (vlr-remove-all)
;;  (vlr-LISP-reactor () '((:vlr-LispWillStart . INSNAME-1)
;;                         (:vlr-LispEnded . INSNAME-2)))
;;  (vlr-COMMAND-reactor () '((:vlr-CommandWillStart . INSNAME-3)
;;                            (:vlr-CommandEnded . INSNAME-4)))
;;
;;  (defun INSNAME-1 (N-REACTIU L-ELISP)
;;    (if (= (substr (car L-ELISP) 1 17) "(C:INSERTOK)") (setq *INSNAME* 0)))
;;
;;  (defun INSNAME-2 (N-REACTIU L-ELISP)
;;    (setq *INSNAME* (if (= *INSNAME* 1) 0)))
;;
;;  (defun INSNAME-3 (N-REACTIU L-ORDRE)
;;    (if (= *INSNAME* 0) (setq *INSNAME* 1)))
;;
;;  (defun INSNAME-4 (N-REACTIU L-ORDRE)
;;    (if (= *INSNAME* 1)
;;        (setq *INSNAME* 0)
;;        (if (= *INSNAME* 0)
;;            (progn
;;              (setq *INSNAME* ())
;;              (setvar "INSNAME" *BLOC*)))))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
```

```
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun INSERT* (/ X* X** Y** OX* OX** BLOC* BL*EX)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0))) (if (< I 0) "-" "+"))
        J (1- (strlen BLOC*)))
  (tblnext "BLOCK" T)
  (while (and (setq BL*EX (tblnext "BLOCK"))
              (/= (substr (cdr (assoc 2 BL*EX)) 1 J) (substr BLOC* 1 J))))
  (if BL*EX
      (setq BLOC* (cdr (assoc 2 BL*EX))
            J (* I (if (= (substr BLOC* (1+ J)) "-")
                       -1 1)))
      (progn
        (setq J 1 K (/ OX* OX**))
        (command "CLAYER" "0"
                 "CECOLOR" "PORBLOQUE"
                 "CELTYPE" "PORBLOQUE"
                 "CELWEIGHT" -2
                 "ATTREQ" 0
                 "INSERT" BLOC O K (* K I) X*
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast)
                 "SCP" "Z" (setq K (trans O 1 0) O O) X**
                 "BLOQUE" BLOC* (trans K 0 1) "P" ""
                 "SCP" "PR"
                 "ATTREQ" (if ATREQ 1 0)
                 "CELWEIGHT" GLIN
                 "CELTYPE" TLIN
                 "CECOLOR" COL
                 "CLAYER" CAPA)))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC N O X Y
                     OX OY XY A B W I J K)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (setq PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
```

```
        O (if (wcmatch O (strcat "*_" N "[+-]"))     ;; Alternativa als reactius
              (substr O 1 (- (strlen O) (strlen N) 2)) O)      ;; LISP i COMMAND.
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
;;        *BLOC* BLOC
        O (getpoint "\nPrecise punto de inserción: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC " <1>: "))
              W (angle O A)
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
  (setvar "ATTDIA" 0)
  (if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
      (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
          (command "INSERT" BLOC O OX (* OY I) X)
          (INSERT*)))
  (prompt (if (> (getvar "CMDACTIVE") 0) "\nIndique valores de atributo\n" "\n"))
  (setvar "ATTDIA" ATDIA)
  (setvar "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))
```

En aquesta versió ja no es dóna l'extemporània repetició del missatge *Verificar valores de atributos*, en desaparèixer **ATRIBS-1** (després **VAL-ATRIBS**) i la prematura assignació de valors d'atribut que aquesta funció orquestrava: ara, quan a **BLOC** hi hagi atributs verificables, l'avís només sortirà precedint les sol·licituds de confirmació. Tanmateix, han aparegut altres deficiències: la inversió d'obliqüitat en els caràcters dels atributs no constants, quan la inserció de **BLOC** continguda a **BLOC*** s'hagi fet amb $\mathbf{E_y < 0}$, i el desplaçament d'aquests atributs, quan el sistema d'ubicació adoptat no faci referència a la línia base. Però deixarem aquests temes per a més endavant, perquè a la VERSIÓ 7 el problema principal és d'ordre pràctic: com que definim **BLOC*** a partir de les restes de l'explosió, tornarem a manipular (amb unes mides i orientació diferents, és clar) els components originals de **BLOC**, i si això ja ens va bé pel que fa als atributs (que hauran recuperat l'estat de preassignació i estaran a punt per rebre en ***-INSERT*** **BLOC*** els valors definitius, sense que ***-ATREDIT*** hagi d'intervenir), hem de reconèixer que **BLOC*** s'ha engreixat; en la seva definició ja no hi figura una referència a **BLOC** sinó una descripció exhaustiva dels seus components, similar a la de **BLOC**. En relació al valor $\frac{O-X''}{O-X'}$ del cosinus de l'angle girat per **BLOC**, la discusió de si aquesta VERSIÓ 7, tot i ser dolenta (per a cada valor sols hi ha un genèric **BLOC***, però el nombre i tipus de components són els mateixos de **BLOC**), ho és menys que la VERSIÓ 3 (on per a cada valor hi havia tants genèrics **BLOC*** com col·leccions diferents de valors d'atribut entre les insercions realitzades, tot i que cadascun d'ells contingués una inserció de **BLOC**, sempre la mateixa llevat del text i altres paràmetres dels atributs no constants) caldria dirimir-la cas per cas.

Com que des de bon començament ja ens havíem pronunciat contra sistemes d'elevada redundància, hauríem d'anar cap a una sol·lució de compromís on s'evités repetir la descripció de **BLOC** en cada genèric **BLOC***. Per exemple, creant una rèplica de **BLOC** la primera vegada que aquest bloc fos utilitzat des de **C:INSERTOK**, rèplica no

materialitzada en un únic bloc sinó en dos, entre els quals es distribuirien els clons dels components originals de **BLOC**:
- **BLOC-1**, on anirien a parar tots els elements gràfics (incloent-hi insercions d'altres blocs), textos i atributs constants i predefinits. Li posarem per nom <nom_de_**BLOC**>_SENSE_ATRIBS.
- **BLOC-2**, on anirien a parar tots els atributs normals (variables i de valor no predefinit). L'anomenarem ATRIBS_DE_<nom_de_**BLOC**>.

La funció **SEGR-ATRIBS** portarà a terme aquesta segregació, creant **BLOC-1** i **BLOC-2**. En comptes d'inserir **BLOC** per constituir cada **BLOC*** genèric, inserirem **BLOC-1** i **BLOC-2** amb els mateixos paràmetres, i tot seguit explosionarem la segona inserció amb ***DESCOMP***, per restituir els atributs (que hauran perdut els valors arbitraris assignats, però que ara ja tindrem escalats, reubicats i girats de manera adient) al seu primitiu estat de preassignació. Així doncs, cada **BLOC*** estarà constituït per una inserció de **BLOC-1** i pels atributs resultants d'explosionar **BLOC-2** (els atributs de **BLOC**, ubicats com ho estarien si haguéssim inserit aquest bloc però reflotats fins al primer nivell), que sol·licitaran valors d'assignació quan executem ***-INSERT***. Però abans de passar al codi, aclarim que només generarem un **BLOC-1** i un **BLOC-2** per a cada bloc primitiu **BLOC** amb atributs no constants ni predefinits: si no n'hi ha cap, treballarem directament amb **BLOC**; si **BLOC** no conté més objectes que atributs normals, no n'hi haurà prou amb un **BLOC-2** idèntic a **BLOC** (que podríem estalviar-nos, jugant directament amb **BLOC** com en el cas precedent, però que crearem per restringir l'accés en profunditat a la funció **SEGR-ATRIBS**), sinó que també haurem de definir un **BLOC-1**, buit però que servirà com a testimoni del signe (+ o -) del factor d'escala $\mathbf{E_y}$ que va aplicar-se a **BLOC** per crear **BLOC***.

Dintre de **SEGR-ATRIBS**, la funció **MAKEBLOC** s'encarregarà de crear **BLOC-1** i **BLOC-2** quan calgui, però no recorrent a l'ordre ***BLOQUE*** sinó a **entmake**. En relació a l'ús de **entmake**, haurem de fer front a una peculiaritat d'aquesta funció predefinida: en la base de dades d'AutoCAD, la descripció dels objectes omet les subllistes de codi 62 (color), 6 (tipus de línia) i 370 (gruix de línia) si aquestes propietats tenen el valor **"PORCAPA"** (**-1** en l'última), però **entmake**, curiosament, no respecta les absències ni les cobreix amb subllistes on s'expliciti aquest valor, sinó que hi posa els valors actuals, respectivament representats per les variables ***CECOLOR***, ***CELTYPE*** i ***CELWEIGHT***, amb les previsibles distorsions; així doncs, abans d'accedir a **MAKEBLOC** donarem el valor **"PORCAPA"** a les tres propietats, perquè els components de **BLOC** amb alguna d'aquestes característiques siguin clonats amb tota fidelitat.

SEGR-ATRIBS aprofitarà l'existència de la variable **BL*EX** (que passarà de local en **INSERT*** a global) per detectar si **BLOC** ja disposa d'algun genèric **BLOC*** (és a dir, si aquest bloc primitiu ha estat utilitzat alguna altra vegada per **C:INSERTOK** en insercions no ortogonals), abans de reutilitzar-la per certificar l'existència o no d'un genèric **BLOC*** concret (referit a un determinat valor $\frac{O\text{–}X''}{O\text{–}X'}$), i així podrem evitar verificacions exhaustives i redefinicions innecessàries de **BLOC-1** i **BLOC-2**. Les variables **BL1EX** i **BL2EX** escatiran l'existència prèvia de **BLOC-1** i **BLOC-2**, i permetran arribar a una diagnosi correcta. Partint de l'evidència que, si **BLOC** té atributs normals i ja hi ha algun bloc genèric **BLOC***, **BLOC-1** també ha d'existir (no podem haver eliminat **BLOC-1** sense abans esborrar les insercions de tots els genèrics **BLOC*** ni eliminar aquests blocs), les situacions que responguin al perfil **(and BL*EX (or (not BL1EX) BL2EX))** no caldrà que les investiguem, perquè el fet que hi hagi algun **BLOC*** revela que **C:INSERTOK** ja ha estat aplicat a **BLOC** de manera no trivial (inserció no ortogonal) i que, en la primera d'aquestes aplicacions,
- o bé s'havia constatat que **BLOC** no tenia atributs normals (si **BLOC-1** no existeix és que ni ell ni **BLOC-2** no han existit mai),
- o bé s'havia constatat que en tenia i van ser creats els blocs substituts **BLOC-1** i **BLOC-2**, que encara subsisteixen (si existeix **BLOC-2**, que era l'únic que podíem haver eliminat sense afectar **BLOC***, és que també existeix **BLOC-1**).

Només passaran per l'adreçador de la segona meitat de **SEGR-ATRIBS** les situacions que responguin al perfil complementari **(or (not BL*EX) (and BL1EX (not BL2EX))**:

1 Hi ha algun **BLOC***, **BLOC-1** existeix però **BLOC-2** no: **BLOC-2** ha estat destruït (segurament, en fer ***LIMPIA*** i acceptar la destrucció de tots els blocs no usats).
2 No hi ha cap **BLOC***, però **BLOC-1** i **BLOC-2** existeixen: o bé es va interrompre la execució la 1ª vegada que **C:INSERTOK** s'aplicava a **BLOC** de manera no trivial, just després de ser creats **BLOC-1** i **BLOC-2**, o bé aquesta 1ª aplicació va reixir però després van ser destruïdes totes les insercions de **BLOC*** i el propi bloc.
3 No hi ha cap **BLOC*** i només existeix un dels substituts de **BLOC**, **BLOC-1** o **BLOC-2**: com en el supòsit 2, amb l'eliminació addicional de **BLOC-2** o **BLOC-1**.

4 No hi ha cap **BLOC***, i **BLOC-1** i **BLOC-2** tampoc no existeixen: podria ser com en el supòsit 2, amb l'eliminació addicional de **BLOC-1** i **BLOC-2**, però el més probable és que sigui la 1ª vegada que **C: INSERTOK** s'aplica a **BLOC** de manera no trivial.

Observeu que, si per estalviar-nos un **BLOC-2** calcat de **BLOC**, en el cas de blocs compostos exclusivament per atributs normals ens haguéssim limitat al **BLOC-1** que dóna testimoni del signe de $\mathbf{E_y}$, cada vegada que els apliquéssim **C:INSERTOK**, i no sols la primera, hauríem de completar la comprovació (amb un tractament diferent, és clar). Però sobretot no perdeu de vista que la condició d'accés a **SEGR-ATRIBS**, **(= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2)**, només serà una manera d'evitar-lo quan **BLOC** no tingui atributs de cap mena; en cas contrari, ningú no ens assegura *a priori* que els seus atributs siguin normals i que, en conseqüència, li haguem de proporcionar uns substituts **BLOC-1** i **BLOC-2**. Per tractar adequadament el supòsit d'un **BLOC** amb atributs, però exclusivament constants i predefinits, **SEGR-ATRIBS** assignarà d'entrada els noms <nom_de_**BLOC**>_SENSE_ATRIBS i ATRIBS_DE_<nom_de_**BLOC**> a les variables **BL1** i **BL2**, i únicament en el cas d'haver comprovat que **BLOC** tenia atributs normals (o que el bloc anomenat <nom_de_**BLOC**>_SENSE_ATRIBS ja existeix), passarà aquests valors a **BLOC-1** i **BLOC-2**; altrament, assignarà el nom de **BLOC** a la variable **BLOC-1**, perquè el protagonisme sigui assumit pel primer, sense necessitat de dobles. Aquest dispositiu també servirà per afrontar determinades incidències: que **C:INSERTOK** se les vegi amb un **BLOC** inicialment proveït d'atributs però que després n'ha estat desposseït per una redefinició. L'assumpte és més sibil·lí del que pugui semblar a primera vista, per l'anòmala resposta d'AutoCAD en aquest cas (no en el contrari, de bloc sense atributs que es redefineix incorporant-ne): el codi 70 de la definició de **BLOC** a la taula de símbols haurà conservat el valor **2** que acreditava l'existència d'atributs, i només quan la redefinició de **BLOC** sigui anterior a la primera intervenció de **C:INSERTOK** sobre aquest bloc, o quan els atributs perduts en la redefinició siguin constants i predefinits exclusivament, el dispositiu funcionarà correctament; si la redefinició deixés sense atributs un **BLOC** amb atributs normals i ja existís algun genèric **BLOC***, tindríem problemes. Però com que això de les redefinicions de blocs ja és una altra història, que diria Kipling, abans d'embolicar-nos més presentarem el nou codi:

```
; VERSIÓ 8

;;  (vl-load-com)
;;  (vlr-remove-all)
;;  (vlr-LISP-reactor () '((:vlr-LispWillStart . INSNAME-1)
;;                         (:vlr-LispEnded . INSNAME-2)))
;;  (vlr-COMMAND-reactor () '((:vlr-CommandWillStart . INSNAME-3)
;;                            (:vlr-CommandEnded . INSNAME-4)))
;;
;;  (defun INSNAME-1 (N-REACTIU L-ELISP)
;;    (if (= (substr (car L-ELISP) 1 17) "(C:INSERTOK)") (setq *INSNAME* 0)))
;;
;;  (defun INSNAME-2 (N-REACTIU L-ELISP)
;;    (setq *INSNAME* (if (= *INSNAME* 1) 0)))
;;
;;  (defun INSNAME-3 (N-REACTIU L-ORDRE)
;;    (if (= *INSNAME* 0) (setq *INSNAME* 1)))
;;
;;  (defun INSNAME-4 (N-REACTIU L-ORDRE)
;;    (if (= *INSNAME* 1)
;;        (setq *INSNAME* 0)
;;        (if (= *INSNAME* 0)
;;            (progn
;;              (setq *INSNAME* ())
;;              (setvar "INSNAME" *BLOC*)))))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
```

```
            (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
            " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "0" "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun MAKEBLOC (BLOC/2 OO ATRIBS)
  (entmake (list '(0 . "BLOCK")
                 (cons 2 BLOC/2)
                 '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun SEGR-ATRIBS (/ BL1 BL2 BL1EX BL2EX OO-1 OO-2 AT AT-CP LB)
  (setq BL1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BL1)
        BLOC-1 (if BL1EX BL1 BLOC)
        BL2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BL2)
        BLOC-2 (if BL2EX BL2))
  (tblnext "BLOCK" T)
  (while (and (setq LB (tblnext "BLOCK"))
              (not (setq BL*EX (wcmatch (cdr (assoc 2 LB)) N)))))
  (if (or (not BL*EX) (and  BL1EX (not BL2EX)))
      (progn
        (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (cdr (assoc 70 LE)) 10) 0))
                   (setq OO-2 (cons LE OO-2))
                   (setq OO-1 (cons LE OO-1)
                         AT-CP (if AT-CP T AT)))
               (setq E (entnext E)))
        (if OO-2
            (progn
              (setvar "CECOLOR" "PORCAPA")
              (setvar "CELTYPE" "PORCAPA")
              (setvar "CELWEIGHT" -1)
              (if (not BL1EX) (MAKEBLOC (setq BLOC-1 BL1) OO-1 AT-CP))
              (if (not BL2EX) (MAKEBLOC (setq BLOC-2 BL2) OO-2 T))
              (setvar "CECOLOR" COL)
              (setvar "CELTYPE" TLIN)
              (setvar "CELWEIGHT" GLIN))))))

(defun INSERT* (/ BLOC-1 BLOC-2 BLOC* BL*EX E LE SS SA X* X** Y** OX* OX**)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
```

```
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2)
      (SEGR-ATRIBS)
      (setq BLOC-1 BLOC))
  (if (setq BL*EX (tblsearch "BLOCK" BLOC*))
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 BL*EX))))) 0)
                       -1 1)))
      (progn
        (setq J 1 K (/ OX* OX**))
        (command "CLAYER" "0"
                 "CECOLOR" "PORBLOQUE"
                 "CELTYPE" "PORBLOQUE"
                 "CELWEIGHT" -2
                 "INSERT" BLOC-1 O K (* K I) X*)
        (setq SS (ssadd (entlast)))
        (if BLOC-2
            (progn
              (command "ATTREQ" 0
                       "INSERT" BLOC-2 O K (* K I) X*
                       "ATTREQ" (if ATREQ 1 0)
                       "DESCOMP" (entlast))
              (setq SA (ssget "P") K -1)
              (while (setq E (ssname SA (setq K (1+ K))))
                     (ssadd E SS))))
        (command "SCP" "Z" (setq K (trans O 1 0) O O) X**
                 "BLOQUE" BLOC* (trans K 0 1) SS ""
                 "SCP" "PR"
                 "CELWEIGHT" GLIN
                 "CELTYPE" TLIN
                 "CECOLOR" COL
                 "CLAYER" CAPA)))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC N O X Y
                     OX OY XY A B W I J K)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (setq PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
        O (if (wcmatch O (strcat "*_" N))      ;; Alternativa a l'ús de reactius
              (substr O 1 (- (strlen O) (strlen N) 1)) O)      ;; LISP i COMMAND.
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
;;        *BLOC* BLOC
        N (strcat BLOC "_" N)
        O (getpoint "\nPrecise punto de inserción: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC " <1>: "))
              W (angle O A)
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
```

```
(setq A (polar O (+ W PI/2) OY)
      B (polar O (- W PI/2) OY)
      XY (distance X Y)
      I (if (< (distance A X) (distance B X)) -1 1))
(setvar "CMDECHO" 0)
(setvar "OSMODE" 0)
(setvar "ATTDIA" 0)
(if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
    (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
        (command "INSERT" BLOC O OX (* OY I) X)
        (INSERT*)))
(prompt (if (> (getvar "CMDACTIVE") 0) "\nIndique valores de atributo\n" "\n"))
(setvar "ATTDIA" ATDIA)
(setvar "OSMODE" OSN)
(setvar "CMDECHO" ECO)
(princ))
```

El tema de les redefinicions de blocs, que hem deixat apuntat just abans d'oferir aquesta última versió, es mereix que li dediquem una mica d'atenció. En principi, sent **C:INSERTOK** una aplicació que genera i utilitza blocs genèrics **BLOC*** derivats d'un **BLOC** original, diríem que millor no tocar-lo un cop l'haguem començat a usar, perquè els efectes induïts sobre les insercions prèvies dels diversos genèrics són impredictibles i també els resultats de les que fem després de la redefinició, a més de possibles errors d'execució. Tot dependrà de la versió que haguem aplicat, i en el parèntesi que obrim ara considerarem les que hem anomenat VERSIÓ 3, VERSIÓ 7 i VERSIÓ 8, úniques que funcionen. Abans, però, convindrà recordar succintament com va tot això de redefinir blocs proveïts d'atributs.

Com és lògic, les insercions posteriors a la redefinició d'un bloc respondran a aquesta última. Un altre tema és què passa amb les insercions ja realitzades, i és aquí on cal distingir entre dues categories de components de bloc: d'una banda, els atributs normals i els predefinits (que, a partir d'ara, anomenarem atributs N i P); de l'altra, la resta d'objectes de dibuix (atributs constants o C, textos simples i múltiples, insercions d'altres blocs –que poden incloure o no atributs de tota mena- i tots els objectes no textuals). L'efecte d'una redefinició sobre les insercions prèvies es podria resumir dient que:

- Els objectes de la primera categoria (atributs N i P) només s'actualitzen si la redefinició s'ha fet amb ***ATTREDEF*** o ***ADMATRB*** (aquesta última ordre, apareguda a la versió 16 d'AutoCAD, no admet la inclusió de nous atributs); si s'ha fet amb ***BLOQUE*** o ***INSERT***, no s'actualitzen.
- Els objectes de la segona categoria (la resta) s'actualitzen sempre.

Aclarirem què s'entén per actualitzar: si la nova definició de bloc incorpora un objecte nou, aquest apareixerà en les insercions prèvies; si se n'ha suprimit un, hi desapareixerà; els canvis (desplaçaments, girs, etc.) es poden interpretar com a desaparició d'un objecte en l'antic estat, seguida d'una reaparició en el nou, o sigui que es transmetran a les insercions prèvies. Però si parlem d'actualització d'atributs N o P, haurem de matisar que:

- Els nous atributs N apareixen a les insercions prèvies amb el valor per defecte, si en tenen.
- La modificació del valor per defecte d'un atribut N o P no afecta les insercions prèvies (podríem dir que s'ha actualitzat el valor per defecte en el sentit de valor potencial d'assignació, però que les assignacions ja estaven fetes); en aquest cas, assimilar canvi a desaparició seguida de reaparició només és vàlid si els atributs també han canviat d'identificador.

Si ara considerem que una inserció del bloc **B1**, proveït d'atributs N i P, forma part de la definició d'un altre bloc **B2**, és pertinent preguntar-se: si redefinim **B1**, ¿això afectarà les insercions de **B2** en els mateixos termes que afectaria les insercions ordinàries de **B1**? Doncs no, perquè l'abast de la redefinició dependrà no només del vehicle utilitzat sinó del moment en què s'hagi produït:

- Les redefinicions de **B1** fetes amb ***BLOQUE***, ***INSERT*** o ***ATTREDEF*** afecten igual totes les insercions de **B2**, anteriors o posteriors a la redefinició: els atributs N o P no s'actualitzen.
- Les redefinicions de **B1** fetes amb ***ADMATRB*** no afecten de la mateixa manera totes les insercions de **B2**, perquè en les anteriors a la redefinició els atributs N o P no s'actualitzen, però en les posteriors sí. Ara bé, com que havíem convingut a cenyir-nos a ACAD 2000 (versió 15 d'AutoCAD), ignorarem aquesta possibilitat.

Una altra cosa molt diferent seria que redefiníssim **B2** utilitzant una inserció actualitzada de **B1** (o, òbviament, que **B2** s'hagués creat després de redefinir **B1**).

En resum:
- INSERCIÓ SIMPLE:
  - Anterior a la redefinició:
    - Atributs N o P:
      - Redefinició amb ***BLOQUE*** o ***INSERT***: no actualitzats.
      - Redefinició amb ***ATTREDEF*** o ***ADMATRB***: actualitzats.
    - Resta: actualitzada.
  - Posterior a la redefinició: tot actualitzat.
- INSERCIÓ D'INSERCIONS ANTERIORS A LA REDEFINICIÓ:
  - Anterior a la redefinició:
    - Amb ***BLOQUE***, ***INSERT***, ***ATTREDEF*** o ***ADMATRB***:
      - Atributs N o P: no actualitzats.
      - Resta: actualitzada.
  - Posterior a la redefinició:
    - Amb ***BLOQUE***, ***INSERT*** o ***ATTREDEF***:
      - Atributs N o P: no actualitzats.
      - Resta: actualitzada.
    - Amb ***ADMATRB***: tot actualitzat.
- INSERCIÓ D'INSERCIONS POSTERIORS A LA REDEFINICIÓ: Sempre tot actualitzat.

Recordant tot això, serà fàcil d'entendre la resposta diferent de cadascuna de les versions esmentades de **C:INSERTOK**, quan la funció ja ha estat aplicada a **BLOC** i redefinim aquest bloc:

VERSIÓ 3:
- Si **BLOC*** és un genèric creat abans de redefinir **BLOC**, en cap inserció, anterior o posterior a la redefinició, no s'actualitzaran els atributs N o P, perquè la inserció de **BLOC** amb què es va crear **BLOC*** era prèvia a l'esdeveniment esmentat. Només amb ***ADMATRB*** s'actualitzarien aquests atributs en les insercions posteriors i, paradoxalment, en un **BLOC** en què la redefinició hagués afectat atributs P (creant-ne de nous, suprimint-ne o simplement canviant el valor predefinit), això comportaria que diferents insercions d'un mateix genèric (les anteriors i posteriors a la redefinició) exhibissin valors diferents d'atribut, contravenint així les regles de joc d'aquesta versió. Si la variable de sistema ***ATTREQ*** està activada, la redefinició de **BLOC** no podrà contemplar cap modificació d'atributs N que comporti variar-ne el nombre, l'ordre d'aparició en escena o el missatge, perquè en les insercions posteriors de **BLOC*** podria fallar la correspondència entre la definició de **BLOC** (antiga) i la llista **WWAA** de valors d'assignació (basada en la nova definició). En la versió que comentem, no només el factor geomètric sinó la coincidència de valors d'atribut era criteri d'especificitat per poder utilitzar un mateix genèric **BLOC***, raó per la qual una afectació dels atributs N en la redefinició de **BLOC** podria no ser causa d'error en l'execució de **C:INSERTOK**: si els **n** primers valors de **WWAA** (**n** seria el menor dels nombres d'atributs N del **BLOC** antic i nou) divergeixen dels valors homòlegs dels **BLOC***s candidats per raons exclusivament geomètriques, se'n crearà un altre en què la definició actual de **BLOC** i la llista **WWAA** estaran en armonia, i amb això ja ens situem en el supòsit descrit a continuació.
- Si **BLOC*** és un genèric creat després de redefinir **BLOC**, les seves insercions estaran actualitzades.

VERSIÓ 7:
- Si **BLOC*** és un genèric creat abans de redefinir **BLOC**, en cap inserció, anterior o posterior a la redefinició, no s'actualitzarà el resultat de l'aplicació (ni els atributs N o P, ni la resta d'objectes), perquè el seu contingut és producte de l'explosió de l'antic **BLOC** i, per tant, es compon d'objectes ja independents d'aquest bloc (de l'antic i d'allò en què pugui convertir-se).
- Si **BLOC*** és un genèric creat després de redefinir **BLOC**, les seves insercions estaran actualitzades.

VERSIÓ 8:
- Si **BLOC** no té atributs N (únic cas en què prescindirem del tàndem format per **BLOC-1** i **BLOC-2**) la composició de **BLOC*** (no pas el nom) serà idèntica a la de VERSIÓ 3, i per aquesta raó:
  - Si **BLOC*** és un genèric creat abans de redefinir **BLOC**, en cap inserció, prèvia o posterior a la redefinició, no s'actualitzaran els atributs P (amb ***ADMATRB***, s'actualitzarien les insercions posteriors).
  - Si **BLOC*** és un genèric creat després de redefinir **BLOC**, les seves insercions estaran actualitzades.

- Si **BLOC** té atributs N (treballem amb el tàndem format per **BLOC-1** i **BLOC-2**):
  - Si, després de redefinir **BLOC**, no es restaura **BLOC-1** ni **BLOC-2**:
    Tant si **BLOC*** és un genèric creat abans de redefinir **BLOC** com si s'ha creat després, en cap inserció no s'actualitzarà el resultat de l'aplicació (ni els atributs N o P, ni la resta d'objectes), perquè la informació continguda en **BLOC** es reparteix entre **BLOC-1** i **BLOC-2** la primera vegada que li apliquem **C:INSERTOK**, i els blocs substituts no s'actualitzaran mentre no els restaurem, encara que mentrestant **BLOC** hagi sofert una redefinició.
  - Si, després de redefinir **BLOC**, s'ha restaurat **BLOC-1**:
    SEGR-ATRIBS només restaura **BLOC-1** quan aquest bloc ha desaparegut de la taula de símbols, eliminat amb ***LIMPIA***; i, perquè això sigui possible, cal que abans hagin estat esborrades totes les insercions de genèrics **BLOC*** (i totes les insercions de **BLOC-1** que haguem pogut fer al marge de **C:INSERTOK**) i eliminades de la taula de símbols les definicions d'aquests genèrics. Així doncs, estarem parlant d'un genèric creat (o tornat a crear) després de redefinir **BLOC**: els atributs P s'actualitzaran sempre, però els atributs N només s'actualitzaran si també es restaura **BLOC-2**.
  - Si, després de redefinir **BLOC**, només s'ha restaurat **BLOC-2**, els atributs C o P no s'actualitzaran, ni tampoc els objectes que no són atributs. Pel que fa als atributs N:
    - Si **BLOC*** és un genèric creat abans de redefinir **BLOC**, no s'actualitzaran en cap inserció, anterior o posterior a la redefinició, perquè el seu contingut és producte de l'explosió de l'antic **BLOC-2** i, per tant, ja és independent d'aquest bloc (de l'antic i d'allò en què pugui convertir-se).
    - Si **BLOC*** és un genèric creat després de redefinir **BLOC**, les seves insercions estaran actualitzades, perquè l'haurem definit amb els atributs resultants d'explosionar una inserció del **BLOC-2** ja restaurat.

En resum, quan redefinim **BLOC**:
- A la VERSIÓ 3 ja se li veu el llautó, en el sentit de quedar en evidència que ***INSERTOK*** no genera unes insercions simples de **BLOC**, com ***-INSERT***, sinó insercions d'insercions de **BLOC**. Però això només passa a les insercions que aprofiten uns genèrics **BLOC*** creats abans que **BLOC** fos redefinit, perquè els genèrics creats després donen lloc a referències completament actualitzades.
- La VERSIÓ 7 és una mica pitjor pel que fa als genèrics pre-redefinició, però les insercions basades en genèrics post-redefinició encara surten actualitzades.
- La VERSIÓ 8 (tret del cas que no hi hagi atributs N, en què es comporta com la VERSIÓ 2) és la pitjor, sobretot si després de la redefinició de **BLOC** no es restaura **BLOC-1** ni **BLOC-2**: cap inserció no s'actualitzarà, amb independència que el genèric **BLOC*** utilitzat sigui pre-redefinició o post-redefinició.

Així doncs, de la VERSIÓ 8 de **C:INSERTOK**, última ideada, quan **BLOC** té atributs N podem dir-ne dues coses:
- Si no eliminem **BLOC-1** ni **BLOC-2** per forçar-ne la restauració (definits d'acord amb la redefinició de **BLOC**), no hi haurà cap mena d'actualització. Per contra, si eliminem tots dos blocs l'actualització serà completa ... però aquesta acció comporta unes restriccions tan severes (l'eliminació de **BLOC-1** està condicionada a la prèvia eliminació de tots els genèrics **BLOC***) que l'afirmació es queda en pura tautologia: posats a començar de cap i de nou, les 3 versions de **C:INSERTOK** serien igualment bones.
- La gestió de **BLOC-1** i **BLOC-2** pot ser la clau de volta per controlar la situació, posant les insercions de **BLOC*** a recer de qualsevol redefinició accidental de **BLOC** o desbloquejant-les i forçant-ne una actualització també extensiva a les realitzades abans d'una redefinició deliberada. Tot i que caldrà un petit canvi: que, en comptes de limitar la restauració al bloc desaparegut, l'absència d'un d'ells activi la restauració dels dos; d'aquesta manera, l'eliminació de **BLOC-2** (que no està sotmesa a cap restricció) també provocarà la restauració de **BLOC-1**.

Quan vulguem que la redefinició de **BLOC** es propagui a tots els genèrics **BLOC***, en els termes establerts, n'hi haurà prou a eliminar **BLOC-2** amb ***LIMPIA***: aquest serà el desencadenant de la redefinició de **BLOC-1** i **BLOC-2** (que ara afectarà tots dos, encara que sols n'hagi desaparegut un) i de l'accés a la nova funció **REDEF-BLOC*S**, que tindrà per missió forçar la redefinició de tots els genèrics **BLOC*** existents, en base al nou **BLOC-1** i als atributs N resultants d'explosionar el nou **BLOC-2**. Només oferim aquesta funció (en què convé advertir que la cerca de blocs genèrics es reemprèn en el primer que havíem localitzat, **LB**, en comptes de reiniciar-la pel començament de la taula fent **(tblnext "BLOCK" T)**) i el final de **SEGR-ATRIBS**, que és l'únic que difereix de la versió precedent:

```
; fragment de VERSIÓ 9
(defun REDEF-BLOC*S (/ BLOK*)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (setvar "ATTREQ" 0)
  (while LB
         (if (wcmatch (setq BLOK* (cdr (assoc 2 LB))) N)
             (progn
               (setq E (cdr (assoc -2 LB))
                     LE (entget E)
                     J (cdr (assoc 41 LE))
                     K (/ (* (cdr (assoc 50 LE)) 180) PI))
               (command "INSERT" BLOC-1 '(0 0 0) J J K)
               (setq SS (ssadd (entlast)))
               (command "INSERT" BLOC-2 '(0 0 0) J J K
                        "DESCOMP" (entlast))
               (setq SA (ssget "P") K -1)
               (while (setq E (ssname SA (setq K (1+ K))))
                      (ssadd E SS))
               (command "BLOQUE" BLOK* "S" '(0 0 0) SS ""))))
         (setq LB (tblnext "BLOCK")))
  (setvar "ATTREQ" (if ATREQ 1 0))
  (setvar "CLAYER" CAPA))

(defun SEGR-ATRIBS (/ BL1 BL2 BL1EX BL2EX OO-1 OO-2 AT AT-CP LB)
; ..........................................................
        (if OO-2
            (progn
              (setq LE (tblsearch "BLOCK" BLOC)
                    BLOC-1 BL1 BLOC-2 BL2)
              (setvar "CECOLOR" "PORCAPA")
              (setvar "CELTYPE" "PORCAPA")
              (setvar "CELWEIGHT" -1)
              (MAKEBLOC BLOC-1 OO-1 AT-CP)
              (MAKEBLOC BLOC-2 OO-2 T)
              (if BL*EX (REDEF-BLOC*S))
              (setvar "CECOLOR" COL)
              (setvar "CELTYPE" TLIN)
              (setvar "CELWEIGHT" GLIN))))))
```

Si ara sotmetem aquesta versió modificada a l'anàlisi comparativa que veníem fent, referida al comportament de ***INSERTOK*** quan ja ha estat aplicada a **BLOC** i redefinim aquest bloc, comprovarem que s'ha recuperat el nivell de resposta de la VERSIÓ 3:

VERSIÓ 9:

- Si **BLOC** no té atributs N (únic cas en què prescindirem del tàndem format per **BLOC-1** i **BLOC-2**) la composició de **BLOC*** (no pas el nom) serà idèntica a la de VERSIÓ 3, i per aquesta raó:
  - Si **BLOC*** és un genèric creat abans de redefinir **BLOC**, en cap inserció, prèvia o posterior a la redefinició, no s'actualitzaran els atributs P (amb ***ADMATRB***, s'actualitzarien les insercions posteriors).
  - Si **BLOC*** és un genèric creat després de redefinir **BLOC**, les seves insercions estaran actualitzades.
- Si **BLOC** té atributs N (treballem amb el tàndem format per **BLOC-1** i **BLOC-2**):
  - Si, després de redefinir **BLOC**, no es restaura **BLOC-1** ni **BLOC-2**:
    Tant si **BLOC*** és un genèric creat abans de redefinir **BLOC** com si s'ha creat després, en cap inserció no s'actualitzarà el resultat de l'aplicació (ni els atributs N o P, ni la resta d'objectes), perquè la informació continguda en **BLOC** es reparteix entre **BLOC-1** i **BLOC-2** la primera vegada que li apliquem **C:INSERTOK**, i els blocs substituts no s'actualitzaran mentre no els restaurem, encara que mentrestant **BLOC** hagi sofert una redefinició.
  - Si, després de redefinir **BLOC**, s'ha restaurat **BLOC-1** i **BLOC-2** (ara, aquestes restauracions ja no estan subordinades a la inexistència dels blocs i tot el que això comporta, sinó que les podem provocar eliminant intencionadament **BLOC-2**):

- Si **BLOC*** és un genèric creat abans de redefinir **BLOC**, la doble restauració n'haurà permés la redefinició (**BLOC-1** i els atributs trets de **BLOC-2** ja respondran al nou **BLOC**): tindrem que en les insercions prèvies els atributs N o P no s'hauran actualitzat (malgrat que els primers pertanyen directament a **BLOC***, sense que cap inserció de **BLOC-2** faci de mitjancera, la redefinició ha estat feta amb ***-BLOQUE***) però en les posteriors sí. Pel que fa a la resta de components, en tots dos casos s'hauran actualitzat.
- Si **BLOC*** és un genèric creat després de redefinir **BLOC**, totes les insercions sortiran actualitzades.

Havent assolit uns resultats acceptablement satisfactoris en allò que jutjàvem més important, potser és el moment d'abordar algunes qüestions menors que ja havíem detectat en la VERSIÓ 7.

La primera d'aquestes anomalies era la inversió d'obliqüitat en els caràcters dels atributs no constants de **BLOC** quan, la primera vegada que **C:INSERTOK** se les veia amb un paral·lelogram d'acollida d'on resultava un valor $\frac{\mathbf{O-X''}}{\mathbf{O-X'}}$ diferent al dels genèrics existents, el nou genèric **BLOC*** s'obtenia inserint **BLOC** amb un factor d'escala $\mathbf{E_y < 0}$: tant en l'estrena d'aquest nou **BLOC*** com en les seves posteriors insercions, els atributs N i P es mostraven escrits amb uns caràcters en què la inclinació era de signe contrari al que tocava (el dels atributs C o els textos escrits amb la mateixa orientació, si n'hi havia), amb independència del signe del factor $\mathbf{E_y}$ aplicat en aquestes insercions. Generalitzant, com que l'anomalia només afecta els atributs N o P components directes de **BLOC***, en la VERSIÓ 3 no n'afecta cap (els atributs N i P pertanyen a una inserció de **BLOC**), en la VERSIÓ 7 afecta tots els atributs N i P (que són components directes de **BLOC***) i en la VERSIÓ 9 només afecta els atributs N (components directes de **BLOC***, ja que els atributs P pertanyen a una inserció de **BLOC-1**). En la Figura 9.1 podeu apreciar el fenòmen: A) inserció directa d'un **BLOC** proveït de tres atributs, C, N i P, que per fer-ho senzill tenen uns valors (preassignats o assignables per defecte) significatius; B) aplicacions de **C:INSERTOK** en què la superior (VERSIÓ 3, 7 o 9) o la inferior (VERSIÓ 3) ha estat la inserció inaugural del genèric **BLOC***; C) aplicacions de **C:INSERTOK** en què la inferior ha estat la inserció inaugural de **BLOC*** (VERSIÓ 7), i D) aplicacions de **C:INSERTOK** en què la inferior ha estat la inserció inaugural de **BLOC*** (VERSIÓ 9).

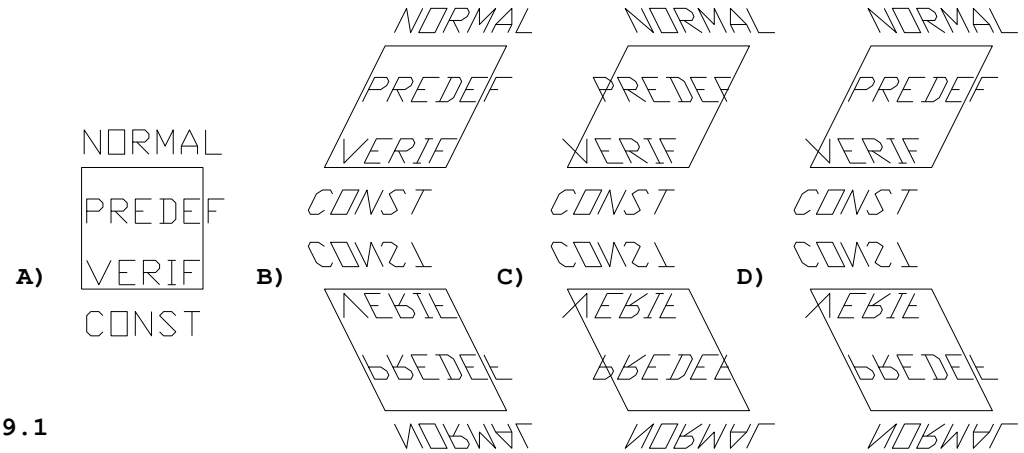


Figura 9.1

Parlant d'inclinació dels caràcters, hem d'aclarir que una cosa són els paràmetres propis de l'estil de text amb què s'ha definit o editat l'atribut (font o tipus de lletra, alçada, factor d'amplada, angle d'obliqüitat i efectes atípics), present a la taula de símbols com a informació pròpia del bloc, i una altra de ben diferent la informació sobre els atributs associada a cada inserció del bloc, on, a més del text assignat, aquests paràmetres ja hi figuren amb els valors concrets adoptats. Encara que els caràcters utilitzats no tinguin obliqüitat, quan un atribut N o P es defineix amb una certa rotació respecte el ***SCP*** del bloc i inserim aquest bloc amb $\mathbf{E_x \neq E_y}$, el codi 51 dels objectes **"ATTRIB"** annexos a l'objecte **"INSERT"** que en resulta expressarà l'obliqüitat real de l'atribut en la inserció, referida a una direcció normal a la d'escriptura. Quan els atributs són ortogonals respecte al

bloc pot passar el mateix si, com a **C:INSERTOK**, inserim el bloc girat i, després de fer de la inserció un altre bloc, inserim aquest amb escalat no uniforme: els caràcters es veuran oblics, tot i que aquesta obliqüitat només serà explícita en la base de dades d'AutoCAD si, mijançant algun artifici (explosió del primer bloc, com fem en VERSIÓ 7 i VERSIÓ 9), alguns dels atributs adquireixen la condició de components directes del segon bloc. En aquest context, la incidència descrita es produirà quan la primera inserció inclogui efecte de simetria, per tenir $\mathbf{E_x}$ i $\mathbf{E_y}$ diferent signe, i en el cas concret de les aplicacions **C:INSERTOK**, quan inserim **BLOC** amb factors d'escala $\mathbf{E_y = -E_x}$: si incorpora atributs C d'una banda, i N i P de l'altra, escrits tots amb el mateix estil i angle de rotació, podrem comprovar com la diferent aparença dels caràcters en cada grup d'atributs es correspon amb el valor invers de l'angle d'obliqüitat associat al codi 51 (com que els angles sempre hi figuren amb valor positiu, la inversió no es percep com a canvi de signe sinó en que restant **360°** a un dels angles s'obté el valor invers de l'altre).

Centrant-nos en la VERSIÓ 9, no és possible introduir una correcció prèvia en cada genèric **BLOC***, consistent a modificar l'obliqüitat dels atributs N (un cop inserit **BLOC-2** i abans d'explosionar aquesta inserció) per contrarestar la inversió que ha d'experimentar en inserir-lo, i la raó és ben senzilla: podríem fer que funcionés en la inserció inaugural i en totes aquelles en què la relació de factors d'escala $\frac{\mathbf{E_x}}{\mathbf{E_y}}$ mantingués el mateix valor (insercions geomètricament semblants); però per a valors diferents, la predeformació seria insuficient o excessiva. No ens quedarà més sortida que corregir l'anomalia *a posteriori*, inserció a inserció, i ho farem invertint l'angle d'inclinació, associat al codi 51 (trobarem el complement a **360°** en comptes de canviar-lo de signe), de tots els atributs (els objectes **"ATTRIB"** annexos al **"INSERT"** de **BLOC*** són atributs N), just al final de la funció **INSERT***. Però, disortadament, la correcció

```
  ..........................................................................
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**)
  (if (< (cdr (assoc 42 (entget (cdr (assoc -2 (tblsearch "BLOCK" BLOC*)))))) 0)
      (progn
        (setq E (entlast))
        (while (setq E (entnext E))
               (setq LE (entget E))
               (if (= (cdr (assoc 0 LE)) "ATTRIB")
                   (progn
                     (entmod (subst (cons 51 (- PI*2 (cdr (assoc 51 LE))))
                                    (assoc 51 LE)
                                    LE))
                     (entupd E)))))))
```

(on la constant **PI*2** valdria **(* PI 2)**, traducció de **360°** a radiants) no pot quedar en un simple afegitó, sinó que hauríem de desempolsar la funció **VAL-ATRIBS** (que no utilitzàvem des de la VERSIÓ 7), per generar la llista **WWAA** de valors d'atributs N i aconseguir que la inserció de **BLOC*** es tanqués amb aquests valors. Altrament, la intervenció de **entmod** seria prematura (no trobaria cap atribut per esmenar-ne la inclinació, perquè **(entnext (entlast))** encara valdria **nil**). Com que l'ús de **WWAA** porta cua (retornar **command** al format de llista sense avaluar i recuperar l'altra vegada operativa assignació **(setvar "INSNAME" BLOC)**), reproduïm el codi sencer:

```
; VERSIÓ 10

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))
```

```
(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun MAKEBLOC (BLOC/2 OO ATRIBS)
  (entmake (list '(0 . "BLOCK")
                 (cons 2 BLOC/2)
                 '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun REDEF-BLOC*S (/ BLOK*)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (setvar "ATTREQ" 0)
  (while LB
         (if (wcmatch (setq BLOK* (cdr (assoc 2 LB))) N)
             (progn
               (setq E (cdr (assoc -2 LB))
                     LE (entget E)
                     J (cdr (assoc 41 LE))
                     K (/ (* (cdr (assoc 50 LE)) 180) PI))
               (command "INSERT" BLOC-1 '(0 0 0) J J K)
               (setq SS (ssadd (entlast)))
               (command "INSERT" BLOC-2 '(0 0 0) J J K
                        "DESCOMP" (entlast))
               (setq SA (ssget "P") K -1)
               (while (setq E (ssname SA (setq K (1+ K))))
                      (ssadd E SS))
               (command "BLOQUE" BLOK* "S" '(0 0 0) SS ""))
         (setq LB (tblnext "BLOCK")))
  (setvar "ATTREQ" (if ATREQ 1 0))
  (setvar "CLAYER" CAPA))

(defun SEGR-ATRIBS (/ BL1 BL2 BL1EX BL2EX OO-1 OO-2 AT AT-CP LB)
  (setq BL1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BL1)
        BLOC-1 (if BL1EX BL1 BLOC)
        BL2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BL2)
        BLOC-2 (if BL2EX BL2))
  (tblnext "BLOCK" T)
  (while (and (setq LB (tblnext "BLOCK"))
              (not (setq BL*EX (wcmatch (cdr (assoc 2 LB)) N)))))
  (if (or (not BL*EX) (and  BL1EX (not BL2EX)))
      (progn
        (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (cdr (assoc 70 LE)) 10) 0))
                   (setq OO-2 (cons LE OO-2))
                   (setq OO-1 (cons LE OO-1) AT-CP (if AT-CP T AT)))
               (setq E (entnext E)))
        (if OO-2
            (progn
              (setq BLOC-1 BL1 BLOC-2 BL2)
              (setvar "CECOLOR" "PORCAPA")
              (setvar "CELTYPE" "PORCAPA")
              (setvar "CELWEIGHT" -1)
```

```
              (MAKEBLOC BLOC-1 OO-1 AT-CP)
              (MAKEBLOC BLOC-2 OO-2 T)
              (if BL*EX (REDEF-BLOC*S))
              (setvar "CECOLOR" COL)
              (setvar "CELTYPE" TLIN)
              (setvar "CELWEIGHT" GLIN))))))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS (/ ICVP M VD V LLAA)
  (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC-2))))
  (while E
         (setq LE (entget E)
               ICVP (cdr (assoc 70 LE)))
         (if (and (not LLAA) ATREQ) (prompt "\nIndique valores de atributo"))
         (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LE))
               (cdr (assoc 3 LE)) (cdr (assoc 1 LE)))
         (setq LLAA (cons (list W N M V) LLAA)
               E (entnext E)))
  (setq W ())
  (foreach LA (reverse LLAA)
    (if (car LA)
        (progn
          (if (and (not W) ATREQ) (prompt "\nVerificar valores de atributo"))
          (VATR (cons "" W) (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq WWAA (append WWAA (list V))))
  (setq WWAA (append WWAA W)))

(defun INSERT* (/ BLOC-1 BLOC-2 BLOC* BL*EX WWAA E LE SS SA X* X** Y** OX* OX**)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2)
      (progn
        (SEGR-ATRIBS)
        (if BLOC-2 (VAL-ATRIBS)))
      (setq BLOC-1 BLOC))
  (if (setq BL*EX (tblsearch "BLOCK" BLOC*))
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 BL*EX))))) 0)
                       -1 1)))
      (progn
        (setq J 1 K (/ OX* OX**))
        (command "CLAYER" "0"
                 "CECOLOR" "PORBLOQUE"
                 "CELTYPE" "PORBLOQUE"
                 "CELWEIGHT" -2
                 "INSERT" BLOC-1 O K (* K I) X*)
        (setq SS (ssadd (entlast)))
        (if BLOC-2
            (progn
              (command "ATTREQ" 0
                       "INSERT" BLOC-2 O K (* K I) X*
                       "ATTREQ" (if ATREQ 1 0)
                       "DESCOMP" (entlast))
```

```
              (setq SA (ssget "P") K -1)
              (while (setq E (ssname SA (setq K (1+ K))))
                     (ssadd E SS))))
       (command "SCP" "Z" (setq K (trans O 1 0) O O) X**
                "BLOQUE" BLOC* (trans K 0 1) SS ""
                "SCP" "PR"
                "CELWEIGHT" GLIN
                "CELTYPE" TLIN
                "CECOLOR" COL
                "CLAYER" CAPA)))
  (eval (append '(command "_INSERT" BLOC* O OX** (* (distance Y** Y) J) X**)
                (if ATREQ WWAA)))
  (if (< (cdr (assoc 42 (entget (cdr (assoc -2 (tblsearch "BLOCK" BLOC*)))))) 0)
      (progn
        (setq E (entlast))
        (while (setq E (entnext E))
               (setq LE (entget E))
               (if (= (cdr (assoc 0 LE)) "ATTRIB")
                   (progn
                     (entmod (subst (cons 51 (- PI*2 (cdr (assoc 51 LE))))
                                    (assoc 51 LE)
                                    LE))
                     (entupd E)))))))

(defun C:INSERTOK (/ Q0 PI*2 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC N O
                     X Y OX OY XY A B W I J K)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (setq PI*2 (* PI 2)
        PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        N (strcat BLOC "_" N)
        O (getpoint "\nPrecise punto de inserción: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC " <1>: "))
              W (angle O A)
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
  (setvar "ATTDIA" 0)
  (if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
      (if (equal (setq J (expt XY 2))
                 (setq K (+ (expt OX 2) (expt OY 2))) Q0)
```

```
        (progn
          (command "INSERT" BLOC O OX (* OY I) X)
          (prompt (if (> (getvar "CMDACTIVE") 0)
                      "\nIndique valores de atributo\n" "\n")))
        (INSERT*)))
  (setvar "INSNAME" BLOC)
  (setvar "ATTDIA" ATDIA)
  (setvar "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))
```

La restauració de la llista **WWAA** no deixa de constituir un pas enrera, que dóna ales a una sospita que planava a l'ambient i no ens podem estar de formular: si tot això només passa quan la necessitat de crear un nou genèric coincideix amb una aplicació de **C:INSERTOK** en què hi ha implícita una simetria en la correspondència geomètrica entre el contingut de **BLOC** i la inserció d'aquest nou **BLOC***, ¿per què, en comptes d'inserir **BLOC** amb $\mathbf{E_{1y} < 0}$ i després inserir-ne el resultat **BLOC*** amb $\mathbf{E_{2y} > 0}$, no hem fet a l'inrevés? Que és possible no hi ha cap dubte, i només cal pensar en els casos representats a la Figura 9.1: l'única diferència entre B i C (VERSIÓ 7) o entre B i D (VERSIÓ 9) és que en el primer cas la inserció inaugural del genèric **BLOC*** no portava implícita cap simetria (inserció superior), mentre que en el segon sí que en portava (insercions inferiors); en definitiva, si en C o D haguéssim efectuat primer les insercions superiors (en què **BLOC*** es construeix sobre una inserció de **BLOC** feta amb $\mathbf{E_{1y} > 0}$, que després s'insereix amb $\mathbf{E_{2y} > 0}$), ni aquestes ni les inferiors haurien presentat cap anomalia (en aquesta última, la inserció de **BLOC*** s'hauria fet amb $\mathbf{E_{2y} < 0}$). Com que tot consistirà a simplificar la funció **INSERT***, eliminant algunes sentències condicionals, restant protagonisme a la variable **J** i realitzant unes mínimes substitucions, presentarem plegades la VERSIÓ 3 i la simplificada VERSIÓ 3+, i subratllarem els canvis en totes dues per poder-los apreciar millor:

```
; fragment de VERSIÓ 3
(defun INSERT* (/ X* X** Y** OX* OX** BLOC* BL*EX E LE)
  (setq A (if (< I 0) A B)
        W (if (< J K) (angle A X) (angle X A))
        B (mapcar '/ (mapcar '+ A X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))) J 1)
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) (ATRIBS-1))
  (if WWAA
      (ATRIBS-2)
      (setq BL*EX (tblsearch "BLOCK" BLOC*)))
  (if BL*EX
      (setq J (* I (if (< (cdr (assoc 42 (entget (cdr (assoc -2 BL*EX))))) 0)
                       -1 1)))
      (eval (append '(command "CLAYER" "0"
                      "CECOLOR" "PORBLOQUE"
                      "CELTYPE" "PORBLOQUE"
                      "CELWEIGHT" -2
                      "INSERT" BLOC O (setq K (/ OX* OX**)) (* K I) X*)
                    (if ATREQ WWAA)
                    '("SCP" "Z" (setq K (trans O 1 0) O O) X**
                      "BLOQUE" BLOC* (trans K 0 1) "LT" ""
                      "SCP" "PR"
                      "CELWEIGHT" GLIN
                      "CELTYPE" TLIN
                      "CECOLOR" COL
                      "CLAYER" CAPA))))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) J) X**))
```

```
; fragment de VERSIÓ 3+
(defun INSERT* (/ X* X** Y** OX* OX** BLOC* BL*EX)
  (setq W (if (< J K) (angle B X) (angle X B))
        B (mapcar '/ (mapcar '+ B X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) (ATRIBS-1))
  (if WWAA
      (ATRIBS-2)
      (setq BL*EX (tblsearch "BLOCK" BLOC*)))
  (if (not BL*EX)
      (eval (append '(command "CLAYER" "0"
                      "CECOLOR" "PORBLOQUE"
                      "CELTYPE" "PORBLOQUE"
                      "CELWEIGHT" -2
                      "INSERT" BLOC O (setq K (/ OX* OX**)) K X*)
                    (if ATREQ WWAA)
                    '("SCP" "Z" (setq K (trans O 1 0) O O) X**
                      "BLOQUE" BLOC* (trans K 0 1) "LT" ""
                      "SCP" "PR"
                      "CELWEIGHT" GLIN
                      "CELTYPE" TLIN
                      "CECOLOR" COL
                      "CLAYER" CAPA))))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) I) X**))
```

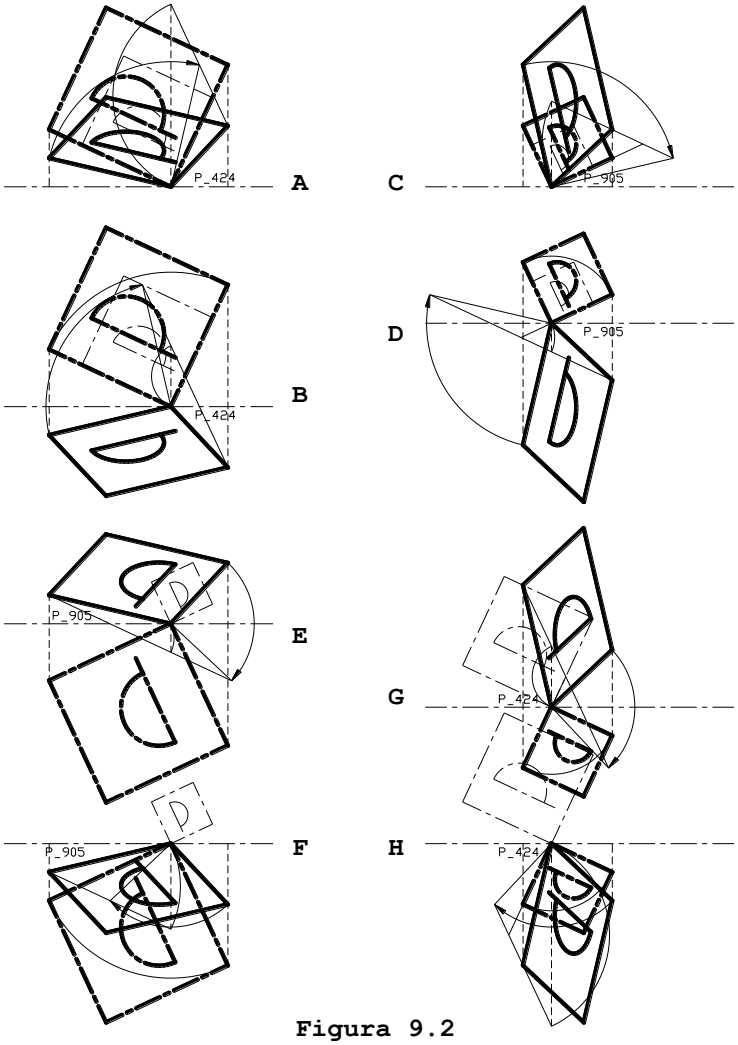


Figura 9.2

La Figura 9.2 ens mostra la traducció gràfica d'aquesta simplificació: el primer pas per obtenir la posició **X'—O—Y'** del quadrat unitari associat a **BLOC**, a partir del paral·lelogram d'acollida **X—O—Y**, és girar **Y 90°** al voltant de **O**, sempre en sentit horari, encara que l'angle format per **O—X** i **O—Y** girat sigui obtús. El resultat és evident, si comparem les construccions A i B, C i D, E i F o G i H: les disposicions de **X—O—Y** simètriques respecte a l'eix d'afinitat **X"—O—Y"** ja no condueixen a **BLOC***s simètrics, que només podíem qualificar d'intercanviables, sinó a **BLOC***s idèntics; és a dir, que tant se val obtenir **BLOC*** d'una inserció en què **X** i **Y** mantinguin la primitiva disposició "mà dreta", com d'una en què la disposició relativa s'hagi transformat en "mà esquerra", perquè el procediment conduirà a un mateix resultat. Quan, en relació a la Figura 8.5, parlàvem dels genèrics A+B+G+H i C+D+E+F, en rigor hauríem hagut de parlar de (A+H)+(B+G) i de (C+F)+(D+E), però amb la simplificació implementada sí que ho podem fer amb absoluta propietat.

Ara ens adonem de la complicació innecessària que havíem introduït, sense ser-ne conscients, en donar entrada a un conveni que fa més còmoda la construcció de Ritz (en situar-se els punts homòlegs a una mateixa banda de l'eix d'afinitat) però que és totalment irrellevant des del punt de vista de l'argumentació geomètrica que desenvolupàvem en el capítol precedent: que el gir inicial de **Y**, **90°** al voltant de **O**, havia de fer-se en aquell sentit (horari, quan la disposició relativa de **X** i **Y** fos "mà dreta", o antihorari, quan fos "mà esquerra") que assegurés que l'angle format per **O—X** i **O—Y** girat seria sempre menor que **90°**.

Tot i que cal entroncar amb la VERSIÓ 9 (la VERSIÓ 10 sols servia per posar pegats als esquinços provocats per l'aplicació del conveni que deixem enrera), veiem les simplificacions addicionals que l'adopció del nou conveni hagués comportat en les versions que s'han anat succeint en el present capítol:

- Si VERSIÓ 4 i VERSIÓ 5 haguessin estat viables, en **INSERT***, a més de les esmenes recollides en VERSIÓ 3+, ens hauríem estalviat les sentències
  (if (< I 0) (command "SCP" "X" "180")) i
  (if (< I 0) (command "SCP" "PR"))
  prèvia i posterior a l'execució de ***–ATREDIT***.

- A la VERSIÓ 7 ens podíem haver estalviat complicar amb un signe **"+"** o **"-"** els noms dels blocs genèrics, i **INSERT*** quedaria així:

```
; fragment de VERSIÓ 7+
(defun INSERT* (/ X* X** Y** OX* OX** BLOC* BL*EX)
  (setq W (if (< J K) (angle B X) (angle X B))
        B (mapcar '/ (mapcar '+ B X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
  (if (not (setq BL*EX (tblsearch "BLOCK" BLOC*)))
      (command "CLAYER" "0"
               "CECOLOR" "PORBLOQUE"
               "CELTYPE" "PORBLOQUE"
               "CELWEIGHT" -2
               "ATTREQ" 0
               "INSERT" BLOC O (setq K (/ OX* OX**)) K X*
               "ATTREQ" (if ATREQ 1 0)
               "DESCOMP" (entlast)
               "SCP" "Z" (setq K (trans O 1 0) O O) X**
               "BLOQUE" BLOC* (trans K 0 1) "P" ""
               "SCP" "PR"
               "ATTREQ" (if ATREQ 1 0)
               "CELWEIGHT" GLIN
               "CELTYPE" TLIN
               "CECOLOR" COL
               "CLAYER" CAPA))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) I) X**))
```

- A la VERSIÓ 8 hauríem pogut simplificar el repertori, prescindint de **BLOC-1** i **BLOC-2** quan **BLOC** no contingués més objectes que atributs normals, en no tenir ja sentit un **BLOC-1** buit, que només es justificava com a testimoni del signe de $\mathbf{E_y}$ (ara serà sempre $\mathbf{E_y > 0}$), el paper de **BLOC-2** el podria assumir **BLOC** directament. Però ens limitarem a polir **INSERT*** per adaptar-la al nou plantejament geomètric

```
; fragment de VERSIÓ 8+
(defun INSERT* (/ BLOC-1 BLOC-2 BLOC* BL*EX E LE SS SA X* X** Y** OX* OX**)
  (setq W (if (< J K) (angle B X) (angle X B))
        B (mapcar '/ (mapcar '+ B X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2)
      (SEGR-ATRIBS)
      (setq BLOC-1 BLOC))
  (if (not (setq BL*EX (tblsearch "BLOCK" BLOC*)))
      (progn
        (command "CLAYER" "0"
                 "CECOLOR" "PORBLOQUE"
                 "CELTYPE" "PORBLOQUE"
                 "CELWEIGHT" -2
                 "INSERT" BLOC-1 O (setq K (/ OX* OX**)) K X*)
        (setq SS (ssadd (entlast)))
        (if BLOC-2
            (progn
              (command "ATTREQ" 0
                       "INSERT" BLOC-2 O K (* K I) X*
                       "ATTREQ" (if ATREQ 1 0)
                       "DESCOMP" (entlast))
              (setq SA (ssget "P") K -1)
              (while (setq E (ssname SA (setq K (1+ K))))
                     (ssadd E SS))))
        (command "SCP" "Z" (setq K (trans O 1 0) O O) X**
                 "BLOQUE" BLOC* (trans K 0 1) SS ""
                 "SCP" "PR"
                 "CELWEIGHT" GLIN
                 "CELTYPE" TLIN
                 "CECOLOR" COL
                 "CLAYER" CAPA)))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) I) X**))
```

  i anirem de dret a revisar la VERSIÓ 9, on sí que adoptarem aquest criteri.

A partir d'ara, només si **BLOC** té atributs N i components de qualsevol altre tipus, crearem **BLOC-1** i **BLOC-2**. Si no té cap atribut N (això ja era així en les versions 8, 9 i 10) o si tots els components ho són (aquesta és la novetat que incorporarem a la revisió, que en atenció a aquesta circumstància anomenarem VERSIÓ 11+ en lloc de VERSIÓ 9+), treballarem directament amb **BLOC**. En conseqüència, en la revisió de **SEGR-ATRIBS** caldrà substituir la condició d'accés a la creació o redefinició de **BLOC-1** i **BLOC-2**, que passarà de ser **(if OO-2 ...)** a ser **(if (and OO-1 OO-2 ...)**, per bé que articulada d'una altra manera. Però no n'hi haurà prou, perquè allò que era una evidència (si **BLOC** té atributs N i hi ha algun **BLOC***, també ha d'existir **BLOC-1**) ja no ho és (si **BLOC** es compon exclusivament d'atributs N, mai no hi haurà **BLOC-1** ni **BLOC-2**), i encara que segueix sent vàlid apartar d'una investigació en profunditat situacions que responen al perfil **(and BL*EX (or (not BL1EX) BL2EX))**, que revela que **C:INSERTOK** ja ha estat aplicat a **BLOC** de manera no trivial i que, en la primera d'aquestes aplicacions,

- o bé s'havia constatat que **BLOC** no tenia atributs N o que tots els components ho eren (si **BLOC-1** no existeix és que ni ell ni **BLOC-2** no han existit mai),
- o bé s'havia constatat que tenia atributs N coexistint amb d'altres components, i van ser creats **BLOC-1** i **BLOC-2**, que encara subsisteixen (si existeix **BLOC-2**, l'únic que podíem haver eliminat sense afectar **BLOC***, també existeix **BLOC-1**),

prèviament haurem de referir-nos al primer subconjunt, que respon a la situació **(and BL*EX (not BL1EX))**, per diferenciar entre els uns (n'hi haurà prou a detectar

que el primer component de **BLOC** no és cap atribut N, i farem **(setq BLOC-1 BLOC)**) i els altres (n'hi haurà prou a detectar que ho és, i farem **(setq BLOC-2 BLOC)**), raó per la qual cal pautar l'anàl·lisi d'una altra manera: en comptes d'encetar-la amb **(or (not BL*EX) (and BL1EX (not BL2EX))**, perfil complementari de l'abans esmentat, ho farem amb **(and BL*EX (not BL1EX))**; així, només quan s'acompleixi la condició **(not (and BL*EX BL2EX))** sobre el seu perfil complementari **(or (not BL*EX) BL1EX)**, proseguirem la investigació en profunditat*. En definitiva, la primera vegada que **C:INSERTOK** manipuli **BLOC** (encara no hi haurà cap genèric **BLOC***), l'analitzarà en profunditat (per crear **BLOC-1** i **BLOC-2**, si s'escau). A partir d'aquest primer cop, sols ho tornarà a fer quan trobi **BLOC-1** però no **BLOC-2** (per haver estat eliminat): d'aquesta manera evitarem perdre el temps amb els **BLOC**s ja processats.

A més de l'adaptació de **SEGR-ATRIBS** a aquests canvis, el codi que presentem inclou petites modificacions en **INSERT***: d'una banda, la simplificació que suposa adoptar un factor d'escala $\mathbf{E_y > 0}$ en inserir **BLOC** per anar definint els diversos genèrics **BLOC*** (en la línia del que havíem vist en VERSIÓ 3+ i VERSIÓ 7+); de l'altra, cal considerar que les variables **BLOC-1** i **BLOC-2** no sempre emmagatzemen els noms d'uns blocs reals, diferents de **BLOC**, sinó que poden haver sofert assignacions nominals excloents (el nom de **BLOC** assignat a **BLOC-1** o **BLOC-2** i, de retruc, **BLOC-2** o **BLOC-1** fora de servei, circumstància la segona que no es donava en VERSIÓ 8 ni VERSIÓ 9). Ens limitarem a reproduir **SEGR-ATRIBS** i **INSERT***, on es concentren tots els canvis:

```
; fragments de VERSIÓ 11+
(defun SEGR-ATRIBS (/ BL1 BL2 BL1EX BL2EX OO-1 OO-2 AT AT-CP LB)
  (setq BL1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BL1)
        BLOC-1 (if BL1EX BL1)
        BL2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BL2)
        BLOC-2 (if BL2EX BL2))
  (tblnext "BLOCK" T)
  (while (and (setq LB (tblnext "BLOCK"))
              (not (setq BL*EX (wcmatch (cdr (assoc 2 LB)) N)))))
  (if (and BL*EX (not BL1EX))
      (progn
        (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
              LE (entget E))
        (if (and (= (cdr (assoc 0 LE)) "ATTDEF")
                 (= (logand (cdr (assoc 70 LE)) 10) 0))
            (setq BLOC-2 BLOC)
            (setq BLOC-1 BLOC)))
      (if (not (and BL*EX BL2EX))
          (progn
            (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
            (while E
                   (setq LE (entget E))
                   (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                            (= (logand (cdr (assoc 70 LE)) 10) 0))
                       (setq OO-2 (cons LE OO-2))
                       (setq OO-1 (cons LE OO-1)
                             AT-CP (if AT-CP T AT)))
                   (setq E (entnext E)))
            (if OO-2
                (if OO-1
                    (progn
                      (setq BLOC-1 BL1 BLOC-2 BL2)
                      (setvar "CECOLOR" "PORCAPA")
                      (setvar "CELTYPE" "PORCAPA")
                      (setvar "CELWEIGHT" -1)
                      (MAKEBLOC BLOC-1 OO-1 AT-CP)
                      (MAKEBLOC BLOC-2 OO-2 T)
                      (if BL*EX (REDEF-BLOC*S))
                      (setvar "CECOLOR" COL)
```

* **(or (not BL*EX) BL1EX)** contempla totes les situacions **(or (not BL*EX) (and BL1EX (not BL2EX))**, a més de **(and BL*EX BL1EX BL2EX)**. Com que aquesta última representa el segon subconjunt i es pot reduir a **(and BL*EX BL2EX)** per tot el que s'ha dit (si existeix **BLOC*** i **BLOC-2**, també existeix **BLOC-1**), per definir l'àmbit d'investigació en profunditat ens limitarem a segregar-la amb **(not (and BL*EX BL2EX))**.

```
                    (setvar "CELTYPE" TLIN)
                    (setvar "CELWEIGHT" GLIN))
                  (setq BLOC-2 BLOC))
                (setq BLOC-1 BLOC))))))

(defun INSERT* (/ BLOC-1 BLOC-2 BLOC* BL*EX E LE SS SA X* X** Y** OX* OX**)
  (setq W (if (< J K) (angle B X) (angle X B))
        B (mapcar '/ (mapcar '+ B X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
  (if (= (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2)
      (SEGR-ATRIBS)
      (setq BLOC-1 BLOC))
  (if (not (setq BL*EX (tblsearch "BLOCK" BLOC*)))
      (progn
        (setq K (/ OX* OX**))
        (command "CLAYER" "0"
                 "CECOLOR" "PORBLOQUE"
                 "CELTYPE" "PORBLOQUE"
                 "CELWEIGHT" -2)
        (if BLOC-1 (command "INSERT" BLOC-1 O K K X*))
        (setq SS (if BLOC-1 (ssadd (entlast))(ssadd)))
        (if BLOC-2
            (progn
              (command "ATTREQ" 0
                       "INSERT" BLOC-2 O K K X*
                       "ATTREQ" (if ATREQ 1 0)
                       "DESCOMP" (entlast))
              (setq SA (ssget "P") K -1)
              (while (setq E (ssname SA (setq K (1+ K))))
                     (ssadd E SS))))
        (command "SCP" "Z" (setq K (trans O 1 0) O O) X**
                 "BLOQUE" BLOC* (trans K 0 1) SS ""
                 "SCP" "PR"
                 "CELWEIGHT" GLIN
                 "CELTYPE" TLIN
                 "CECOLOR" COL
                 "CLAYER" CAPA)))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) I) X**))
```

Igual que fèiem a la VERSIÓ 9, quan vulguem que la redefinició de **BLOC** es propagui a tots els genèrics **BLOC*** n'hi haurà prou a eliminar **BLOC-2** amb ***LIMPIA***, i aquest serà el desencadenant de la redefinició de **BLOC-1** i **BLOC-2**, però ara això només es produeix quan els atributs N coexisteixen amb d'altres components, raó per la qual cal completar l'anàlisi comparativa, incorporant-hi la resposta de **C:INSERTOK** (si ja l'havíem aplicat a **BLOC** abans de redefinir-lo) corresponent a la versió actual:

VERSIÓ 11+:

- Si **BLOC** no té atributs N o tots els components ho són (casos en què prescindim del tàndem format per **BLOC-1** i **BLOC-2**), la present versió es comportarà com la VERSIÓ 3, pel que fa als atributs P (primer cas), o com la VERSIÓ 7, pel que fa als atributs N (segon cas). Així doncs:
  - Si **BLOC*** és un genèric creat abans de redefinir **BLOC**, en cap inserció, prèvia o posterior a la redefinició, no s'actualitzaran els atributs N o P (***ADMATRB*** podria actualitzar els atributs P en les insercions posteriors).
  - Si **BLOC*** és un genèric creat després de redefinir **BLOC**, les seves insercions estaran actualitzades.
- Si **BLOC** té atributs N però també components d'altre tipus (aquest és l'únic cas en què treballem amb el tàndem format per **BLOC-1** i **BLOC-2**):
  - Si, després de redefinir **BLOC**, no es restaura **BLOC-1** ni **BLOC-2**:
    Tant si **BLOC*** és un genèric creat abans de redefinir **BLOC** com si s'ha creat després, en cap inserció no s'actualitzarà el resultat de l'aplicació (ni els

atributs N o P, ni la resta d'objectes), perquè la informació continguda en **BLOC** es reparteix entre **BLOC-1** i **BLOC-2** la primera vegada que li apliquem **C:INSERTOK**, i els blocs substituts no s'actualitzaran mentre no els restaurem, encara que mentrestant **BLOC** hagi sofert una redefinició.

- Si, després de redefinir **BLOC**, s'ha restaurat **BLOC-1** i **BLOC-2** (restauracions que ja no estan subordinades a la inexistència dels blocs i tot el que això comporta, sinó que podem provocar eliminant intencionadament **BLOC-2**):
  - Si **BLOC*** és un genèric creat abans de redefinir **BLOC**, la doble restauració n'haurà permés la redefinició (**BLOC-1** i els atributs trets de **BLOC-2** ja respondran al nou **BLOC**): tindrem que en les insercions prèvies els atributs N o P no s'hauran actualitzat (malgrat pertànyer directament a **BLOC***, sense que cap inserció de **BLOC-2** faci de mitjancera, la redefinició ha estat feta amb ***-BLOQUE***) però en les posteriors sí. Pel que fa a la resta de components, en tots dos casos s'hauran actualitzat.
  - Si **BLOC*** és un genèric creat després de redefinir **BLOC**, totes les insercions sortiran actualitzades.

El balanç és regressiu en relació a la VERSIÓ 9 que hem revisat, pel que fa als blocs compostos exclusivament per atributs N. Però allò que produeix més malestar és l'artificiositat de la frontera que separa els blocs compostos en exclusiva per atributs N o per objectes que no ho són d'aquells altres blocs en què coexisteixen ambdues menes de components. Ho justificàvem parlant d'economia de recursos, però fa de mal entendre que un bloc compost exclusivament per atributs N hagi de passar al segon grup (amb el diferent tracte que això representa) si li afegim una línia o si substituïm un atribut N per un atribut P; o que un de compost exclusivament per línies o per atributs P, si li afegim un atribut N o si substituïm un atribut P per un atribut N. De fet, redefinicions com aquestes o com les contràries, que suposen una mínima modificació del bloc però que comporten un canvi de lloc a tan arbitrària classificació, no estan contemplades en la VERSIÓ 11+: perquè no hi ha possibilitat de forçar la redefinició, propagant-la de **BLOC** a **BLOC*** (quan partim del primer grup no disposem de cap **BLOC-2** per eliminar, i quan partim del segon grup l'eliminació de **BLOC-2** no arrossega la de **BLOC-1**), o perquè la supressió o l'addició d'atributs N provoquen errors d'execució. A tall d'exemple, considerem que **BLOC** no tenia atributs N i que, després d'haver-lo utilitzat amb **C:INSERTOK**, decidim redefinir-lo incorporant-hi atributs N i seguim utilitzant-lo en aquesta aplicació. La disparitat de resultats dependrà d'un factor tan aleatori com la manera en què haguem seleccionat els diversos components del nou **BLOC** a l'hora de redefinir-lo mitjançant ***BLOQUE*** i, al capdavall, de l'ordre en què apareguin en escena. En efecte, com que hem creat genèrics **BLOC*** però no **BLOC-1** ni **BLOC-2**, en **SEGR-ATRIBS** l'acompliment de la condició **(and BL*EX (not BL1EX))** provocarà que el nom de **BLOC** sigui assignat a la variable **BLOC-2** si el primer component localitzat en la nova definició de **BLOC** és un atribut N, o a la variable **BLOC-1** si no ho és. Això no tindrà cap rellevància mentre **C:INSERTOK** treballi amb **BLOC***s existents, però quan n'hagi de crear un de nou començaran a produir-se irregularitats:

- Si ha resultat que **BLOC-2** = **BLOC** i, per omissió, **BLOC-1** = **nil** (que consti que només parlem de variables, ja que cap bloc de nom <nom_de_**BLOC**>_SENSE_ATRIBS o ATRIBS_DE_<nom_de_**BLOC**> ha estat creat), la funció **INSERT*** crearà el nou **BLOC*** amb el producte de la inserció i posterior descomposició de **BLOC**, és a dir que la VERSIÓ 11+ trairà els principis que l'han inspirat i en aquest cas funcionarà com la VERSIÓ 7+. Tret d'això, que no és poc, no es produiran errors d'execució.
- Si ha resultat que **BLOC-1** = **BLOC** i, per omissió, **BLOC-2** = **nil**, en el millor dels casos la VERSIÓ 11+ tampoc no aplicarà els seus principis sinó que farà com la VERSIÓ 3+, definint **BLOC*** amb una inserció de **BLOC** en què els atributs N hauran adquirit valors imprevistos. Per a més inri, la inserció inaugural donarà error i no arribarà a materialitzar-se, i les posteriors, a banda d'exhibir aquests valors no desitjats, tampoc no adaptaran al paral·lelogram d'encaix el quadrat unitari associat a **BLOC**. Per entendre què està passant, haurem de remuntar-nos a les consideracions que ens havien permès de simplificar la VERSIÓ 2, donant lloc a la VERSIÓ 3. Dèiem aleshores que, quan la funció **command** no disposa de prou arguments per enllestir l'execució d'una ordre AutoCAD, els busca entre els de la següent funció **command**, i aquest és l'origen de l'embolic: com que **INSERT*** no contempla la possibilitat que el nom **BLOC-1** sigui el d'un bloc amb atributs N, abans d'inserir-lo no desactiva la variable de sistema ***ATTREQ***; en conseqüència, la inserció inaugural pren com a valors d'assignació els arguments **"SCP"**, **"Z"**, **(setq K (trans 0 1 0) 0 0)** i **X*** de la primera **command** que troba a continuació. Si no hi ha més atributs N que aquests quatre arguments, no li caldrà consumir també **"BLOQUE"**, i **BLOC*** quedarà definit (a la manera de VERSIÓ 3, com hem dit), encara que la primera inserció no prosperi en produir-se l'error *No hay ningún*

sistema de coordenadas guardado ; error: Función cancelada, provocat per un ***SCP PRevio*** a qui se li ha escamotejat el ***SCP Z*** que l'havia de precedir. Però quan n'hi hagi més de quatre ni **BLOC*** no haurà pogut constituir-se.

En vista d'això, haurem d'admetre que la tan pregonada economia de mitjans no era compatible amb l'objectiu d'optimitzar la propagació de qualsevol redefinició de **BLOC** als genèrics **BLOC*** existents (insercions prèvies i noves) o venidors, i que més val que diversifiquem els esforços segons les prioritats: una VERSIÓ 12+ on hi primi l'economia (**BLOC-1** i **BLOC-2** limitats a blocs **BLOC** amb atributs N i d'altres components), no apta per a redefinicions de **BLOC**, i una VERSIÓ 13+, més pròdiga en recursos (**BLOC-1** i **BLOC-2** en tots els casos, encara que un dels dos estigui buit de contingut) i pensada per suportar redefinicions de **BLOC** i traslladar-les a les insercions de **BLOC*** en les mateixes condicions que les insercions simples de **BLOC**. Entre VERSIÓ 12+ i VERSIÓ 13+ se situaria l'oblidada VERSIÓ 8+, no tan "econòmica" com la primera (i, com aquesta, no apta per a redefinicions) però que compartiria amb la segona la possibilitat de superar amb elegància l'últim obstacle pendent de resolució (qüestió aquesta que abordarem en breu), gràcies a haver preservat l'ús de **BLOC-1** i **BLOC-2** quan els **BLOC**s només estaven constituïts per atributs N.

Pel que fa a la VERSIÓ 12+, aclarim que no arrencarem de la VERSIÓ 8 sinó de la VERSIÓ 11+, tot i la voluntat confessada de no contemplar la redefinició de **BLOC**: com en aquesta última, restringirem l'ús de blocs substituts **BLOC-1** i **BLOC-2** als **BLOC**s on els atributs N coexisteixin amb d'altres components, però prescindint de la funció **REDEF-BLOC*S** i fent que **MAKEBLOC** només intervingui, com a la VERSIÓ 8, per crear **BLOC-1** i **BLOC-2**, o per restaurar algun **BLOC-2** eliminat accidentalment. Els canvis respecte a VERSIÓ 11+ es limiten, doncs, a la supressió de **REDEF-BLOC*S** i al tram final de la funció **SEGR-ATRIBS**, que oferim a continuació:

```
; fragments de VERSIÓ 12+
  (defun SEGR-ATRIBS (/ BL1 BL2 BL1EX BL2EX OO-1 OO-2 AT AT-CP LB)
; .............................................................
            (if OO-2
                (if OO-1
                    (progn
                      (setq BLOC-1 BL1 BLOC-2 BL2)
                      (setvar "CECOLOR" "PORCAPA")
                      (setvar "CELTYPE" "PORCAPA")
                      (setvar "CELWEIGHT" -1)
                      (if (not BL1EX) (MAKEBLOC BLOC-1 OO-1 AT-CP))
                      (if (not BL2EX) (MAKEBLOC BLOC-2 OO-2 T))
                      (setvar "CECOLOR" COL)
                      (setvar "CELTYPE" TLIN)
                      (setvar "CELWEIGHT" GLIN))
                    (setq BLOC-2 BLOC))
                (setq BLOC-1 BLOC))))))
```

Abordant ara la VERSIÓ 13+, concebuda per traslladar qualsevol redefinició de **BLOC** a tots els genèrics **BLOC*** al preu de generalitzar la suplantació de **BLOC** per part de **BLOC-1** i **BLOC-2**, encara que un d'ells estigui buit de contingut, convé tornar a la VERSIÓ 9, més pròxima a aquests objectius, per actuar-hi en en tres fronts:

- De primer, adaptar-la a la simplificació geomètrica incorporada en les versions 3+, 7+, 8+, 11+ i 12+ (genèrics **BLOC*** amb la lateralitat no invertida respecte a **BLOC**), és a dir, anar al que anomenaríem VERSIÓ 9+ si ens haguéssim aturat aquí.
- Passar les llistes de components **OO-1** i **OO-2** de variables locals en **SEGR-ATRIBS** a variables locals en **INSERT***, utilitzant-les en substitució de **BLOC-1** i **BLOC-2** (que, en existir ara sempre, ni que sigui com a blocs desproveïts de contingut, són irrellevants com a valors booleans) per identificar la composició de **BLOC**. En realitat, en tindríem prou amb la segona.
- Fer de **SEGR-ATRIBS** una estació de parada obligada, fins i tot per a **BLOC**s sense atributs. Dintre d'aquesta funció, però, la condició d'accés a una investigació en profunditat es simplificarà, ja que només en prescindirem quan s'acompleixi **(and BL1EX BL2EX)**, amb independència del valor **BL*EX** que utilitzarem únicament com a condició d'accés a **REDEF-BLOC*S** (de fet, aquesta funció és autosuficient i no li cal cap control extern suplementari, però mantindrem la intervenció de **BL*EX** per estalviar-nos en el cas **(not BL*EX)** el ball estèril de les variables de sistema ***CLAYER***, ***CECOLOR***, ***CELTYPE*** i ***CELWEIGHT***): quan trobem a faltar **BLOC-1** (destruït fortuïtament, cosa que sols ha pogut passar si és **(not BL*EX)**), **BLOC-2** (destruït de manera fortuïta o intencionadament, per estendre una redefinició de

BLOC a **BLOC-1** i **BLOC-2**, i forçar l'actualització dels **BLOC***s existents) o tots dos a la vegada (per destrucció fortuïta o perquè no han existit mai, en ser aquest el primer cop que **C:INSERTOK** treballa amb **BLOC**), haurem de realitzar la dissecció de **BLOC** per formar amb els seus components les llistes **OO-1** i **OO-2**, reconvertides per **MAKEBLOC** a **BLOC-1** i **BLOC-2**, tant si estan plenes com si no.

Així doncs, pel que fa a la VERSIÓ 13+, el major abast de les reformes efectuades sobre la VERSIÓ 9 i la seva adopció com a punt de partença per seguir el discurs, enfront de la VERSIÓ 12+, aconsellen de reproduir el codi íntegrament:

```
; VERSIÓ 13+

;;  (vl-load-com)
;;  (vlr-remove-all)
;;  (vlr-LISP-reactor () '((:vlr-LispWillStart . INSNAME-1)
;;                         (:vlr-LispEnded . INSNAME-2)))
;;  (vlr-COMMAND-reactor () '((:vlr-CommandWillStart . INSNAME-3)
;;                            (:vlr-CommandEnded . INSNAME-4)))
;;
;;  (defun INSNAME-1 (N-REACTIU L-ELISP)
;;    (if (= (substr (car L-ELISP) 1 17) "(C:INSERTOK)") (setq *INSNAME* 0)))
;;
;;  (defun INSNAME-2 (N-REACTIU L-ELISP)
;;    (setq *INSNAME* (if (= *INSNAME* 1) 0)))
;;
;;  (defun INSNAME-3 (N-REACTIU L-ORDRE)
;;    (if (= *INSNAME* 0) (setq *INSNAME* 1)))
;;
;;  (defun INSNAME-4 (N-REACTIU L-ORDRE)
;;    (if (= *INSNAME* 1)
;;        (setq *INSNAME* 0)
;;        (if (= *INSNAME* 0)
;;            (progn
;;              (setq *INSNAME* ())
;;              (setvar "INSNAME" *BLOC*)))))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun MAKEBLOC (BLOC/2 OO ATRIBS)
  (entmake (list '(0 . "BLOCK")
                 (cons 2 BLOC/2)
                 '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))
```

```
(defun REDEF-BLOC*S (/ BLOK*)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (setvar "ATTREQ" 0)
  (while LB
         (if (wcmatch (setq BLOK* (cdr (assoc 2 LB))) N)
             (progn
               (setq E (cdr (assoc -2 LB))
                     LE (entget E)
                     J (cdr (assoc 41 LE))
                     K (/ (* (cdr (assoc 50 LE)) 180) PI))
               (command "INSERT" BLOC-1 '(0 0 0) J J K)
               (setq SS (ssadd (entlast)))
               (if OO-2
                   (progn
                     (command "INSERT" BLOC-2 '(0 0 0) J J K
                              "DESCOMP" (entlast))
                     (setq SA (ssget "P") K -1)
                     (while (setq E (ssname SA (setq K (1+ K))))
                            (ssadd E SS))))
               (command "BLOQUE" BLOK* "S" '(0 0 0) SS ""))))
         (setq LB (tblnext "BLOCK")))
  (setvar "ATTREQ" (if ATREQ 1 0))
  (setvar "CLAYER" CAPA))

(defun SEGR-ATRIBS (/ BL1EX BL2EX AT AT-CP LB)
  (setq BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2))
  (if (and BL1EX BL2EX)
      (setq OO-2 (/= (cdr (assoc 0 (entget (cdr (assoc -2 BL2EX))))) "ENDBLK"))
      (progn
        (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (cdr (assoc 70 LE)) 10) 0))
                   (setq OO-2 (cons LE OO-2))
                   (setq OO-1 (cons LE OO-1)
                         AT-CP (if AT-CP T AT)))
               (setq E (entnext E)))
        (tblnext "BLOCK" T)
        (while (and (setq LB (tblnext "BLOCK"))
                    (not (setq BL*EX (wcmatch (cdr (assoc 2 LB)) N)))))
        (setvar "CECOLOR" "PORCAPA")
        (setvar "CELTYPE" "PORCAPA")
        (setvar "CELWEIGHT" -1)
        (MAKEBLOC BLOC-1 OO-1 AT-CP)
        (MAKEBLOC BLOC-2 OO-2 OO-2)
        (if BL*EX (REDEF-BLOC*S))
        (setvar "CECOLOR" COL)
        (setvar "CELTYPE" TLIN)
        (setvar "CELWEIGHT" GLIN))))

(defun INSERT* (/ BLOC-1 BLOC-2 BLOC* BL*EX OO-1 OO-2 E LE SS SA X* X** Y**
                  OX* OX**)
  (setq W (if (< J K) (angle B X) (angle X B))
        B (mapcar '/ (mapcar '+ B X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*) OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
```

```
  (SEGR-ATRIBS)
  (if (not (setq BL*EX (tblsearch "BLOCK" BLOC*)))
      (progn
        (command "CLAYER" "0"
                 "CECOLOR" "PORBLOQUE"
                 "CELTYPE" "PORBLOQUE"
                 "CELWEIGHT" -2
                 "INSERT" BLOC-1 O (setq K (/ OX* OX**)) K X*)
        (setq SS (ssadd (entlast)))
        (if OO-2
            (progn
              (command "ATTREQ" 0
                       "INSERT" BLOC-2 O K K X*
                       "ATTREQ" (if ATREQ 1 0)
                       "DESCOMP" (entlast))
              (setq SA (ssget "P") K -1)
              (while (setq E (ssname SA (setq K (1+ K))))
                     (ssadd E SS))))
        (command "SCP" "Z" (setq K (trans O 1 0) O O) X**
                 "BLOQUE" BLOC* (trans K 0 1) SS ""
                 "SCP" "PR"
                 "CELWEIGHT" GLIN
                 "CELTYPE" TLIN
                 "CECOLOR" COL
                 "CLAYER" CAPA)))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) I) X**))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC N O X Y
                     OX OY XY A B W I J K)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (setq PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
        O (if (wcmatch O (strcat "*_" N))      ;; Alternativa a l'ús de reactius
              (substr O 1 (- (strlen O) (strlen N) 1)) O)      ;; LISP i COMMAND.
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
;;        *BLOC* BLOC
        N (strcat BLOC "_" N)
        O (getpoint "\nPrecise punto de inserción: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC " <1>: "))
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
```

```
(setvar "ATTDIA" 0)
(if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
    (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
        (command "INSERT" BLOC O OX (* OY I) X)
        (INSERT*)))
(prompt (if (> (getvar "CMDACTIVE") 0) "\nIndique valores de atributo\n" "\n"))
(setvar "ATTDIA" ATDIA)
(setvar "OSMODE" OSN)
(setvar "CMDECHO" ECO)
(princ))
```

Dels problemes lligats a l'alliberament dels atributs dintre de **BLOC*** (reflotació fins al primer nivell), que en la VERSIÓ 7 afectava els atributs P i N, i en les posteriors només els últims, el d'inversió de l'obliqüitat dels caràcters, quan **BLOC*** es definia sobre una inserció de **BLOC** efectuada amb $\mathbf{E_y < 0}$, ja l'hem resolt en depurar el procediment d'obtenció de paràmetres geomètrics i evitar l'ús de $\mathbf{E_y}$ negatius, amb la VERSIÓ 7+ i a partir de la VERSIÓ 11+. Però quedava per resoldre el desplaçament d'atributs ubicats en relació a les línies superior, mitjana o inferior d'escriptura, i ja és hora que ens hi posem, començant per identificar el problema i escatir-ne les causes.

Quan inserim un bloc que té textos o atributs amb una determinada inclinació (no paral·lels a l'eix **X**) i uns criteris de posicionament no lligats a la línia base (línia superior, mitjana o inferior), i ho fem usant factors d'escala $\mathbf{E_x \neq E_y}$, aquests components queden desplaçats en la direcció de l'escriptura, si bé cal distingir entre atributs N i P d'una banda, i atributs C i textos de l'altra:
- En atributs normals i predefinits, el fenòmen es manifesta tot just realitzada la inserció.
- En atributs constants i textos, cal explosionar la inserció perquè es produeixi el desplaçament.

La resposta diversa obeeix a l'estatus també diferent dels dos grups d'objectes, però de moment ens centrarem en els del segon grup i, en concret, en els textos. Si fem unes quantes proves, jugant amb blocs on anem variant de forma sistemàtica les característiques d'aquests components, ens adonarem que el desplaçament depèn de la distància de la línia de referència adoptada (superior, mitjana o inferior) a la línia base, de la inclinació dels textos i de la relació d'escales $\frac{\mathbf{E_x}}{\mathbf{E_y}}$ de la inserció. Si deixem de banda els modes de justificació lligats a la línia base (*punto inicial*, ***C**entro*, ***D**erecha*, a***J**ustar* i a***L**inear*), anomenem punt d'ancoratge del text l'utilitzat per ubicar-lo en el dibuix i punt base el que sobre la línia base correspon a la mateixa justificació horitzontal (és a dir, *punto inicial* per als punts d'ancoratge ***S**uperior-**IZ**quierda*, ***M**edio-**I**zquierda* o ***I**nferior-**I**zquierda*; ***C**entro* per als punts d'ancoratge ***S**uperior-**C**entro*, ***M**edio-**C**entro* –també ***M**edio*, sense més- o ***I**nferior-**C**entro*; ***D**erecha* per als punts d'ancoratge ***S**uperior-**D**erecha*, ***M**edio-**D**erecha* o ***I**nferior-**D**erecha*), tard o d'hora descobrirem que el vector de desplaçament està determinat per dues posicions: el punt base i la projecció perpendicular del punt d'ancoratge sobre la línea base, cosa que ens posarà sobre la pista del que passa entre bastidors.

Per entendre el fenómen, hauríem de considerar dos textos escrits amb un estil oblic (suposem que utilitza el tipus de lletra *TXT*, amb un angle d'obliqüitat $\boldsymbol{\varphi = -20°}$, per exemple), l'un ancorat a *punto inicial* i l'altre al punt ***SIZ***. Si volem veure'ls perfectament superposats, a més de tenir un mateix contingut (la lletra majúscula "H", per exemple), ¿on creieu que hauran de situar-se els dos punts de justificació: on vulguem que se situï l'extrem superior i el inferior, respectivament, del pal esquerre de la "H"? Si i no: en el primer text és clar que on entrem el punt d'ancoratge se situarà la pota inferior esquerra de la "H", però si volem que el segon se situï al damunt del primer, el punt d'ancoratge ***SIZ*** no l'hem de fer coincidir amb la pota superior esquerra de la primera "H" sinó amb la projecció perpendicular de la pota inferior esquerra sobre la línia superior del text; situant el punt d'ancoratge ***SIZ*** sobre la pota superior esquerra, la segona "H" quedaria desplaçada una longitud $\mathbf{h}\cos\boldsymbol{\varphi}$ (**h** és l'alçada del text, o distància recta entre la línia base i la línia superior) cap a l'esquerra. Tres quarts del mateix passaria aplicant modes de justificació ***MI*** o ***II*** al segon text: hauríem de situar els punts d'ancoratge sobre la línia mitjana o la línia inferior del text, però en la perpendicular traçada des de la pota inferior esquerra de la primera "H" i no pas seguint des d'aquest punt una orientació de **20°** respecte de la normal

(perquè igual que abans, els textos sortirien desplaçats una distància $\frac{h}{2}\cos\varphi$ cap a l'esquerra, en el primer cas, i $\frac{h}{3}\cos\varphi$ cap a la dreta, en el segon).

1 Transformat del punt inicial

2 Punt d'ancoratge que correspon al transformat del punt inicial

3 Transformat del punt d'ancoratge

4 Adoptat com a punt d'ancoratge

5 Punt inicial que correspon a l'adoptat com a punt d'ancoratge

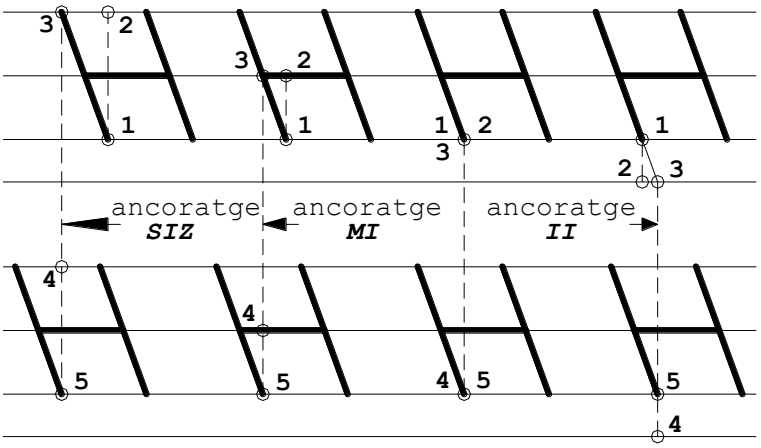


Figura 9.3

Com que aquest criteri és aplicable a qualsevol altra justificació horitzontal, el podríem enunciar dient que els punts ***SIZ***, ***MI***, *punto inicial* i ***II*** sempre s'alineen perpendicularment a la direcció d'escriptura, igual que els punts ***SC***, ***MC*** o ***M***, ***C*** i ***IC***, i que els punts ***SD***, ***MD***, ***D*** i ***ID***, amb independència de l'obliqüitat de l'estil utilitzat. Dit d'una altra manera, a partir de punts d'ancoratge ***SIZ***, ***MI*** o ***II***, el text quedarà igual que usant la projecció sobre la línia base com a *punto inicial*; a partir de punts ***SC***, ***MC*** (o ***M***) o ***IC***, quedarà igual que usant la projecció sobre la línia base com a punt ***C***; i, a partir de punts ***SD***, ***MD*** o ***ID***, quedarà igual que usant la projecció sobre la línia base com a punt ***D***. En definitiva, encara que un text estigui escrit amb caràcters oblics, el seu punt d'ancoratge i la posició que hem anomenat punt base formen angle recte amb la línia base. En realitat, com que les úniques posicions codificades a la base de dades d'AutoCAD són el punt d'ancoratge (codi 11) i el punt inicial (codi 10), que serveix per saber on s'ha de començar a escriure i que només en els casos ***SIZ***, ***MI*** i ***II*** coincideix amb el punt base, filant prim hauríem de dir que, encara que un text estigui escrit amb caràcters oblics, el punt inicial se situarà de manera que el punt base (determinat per l'anterior, si també coneixem la direcció d'escriptura, la longitud del text i la justificació horitzontal) sigui el peu de la perpendicular traçada des del punt d'ancoratge a la línia base.

Quan un text inclinat però escrit amb caràcters rectes ($\varphi = 0°$) i justificat sobre la línia superior, mitjana o inferior, forma part d'un bloc que s'insereix amb factors d'escala $E_x \neq E_y$, malgrat que l'aspecte sigui idèntic al d'un text escrit amb caràcters oblics, perquè l'angle que formen els pals de les lletres amb la línia base ha deixat de ser recte, AutoCAD no es deixa enredar perquè coneix la procedència de l'objecte, en mantenir oberts dos canals d'informació: la definició del bloc (on el text hi figura en les condicions originals, amb caràcters rectes) i aquesta referència concreta al bloc (l'objecte **"INSERT"**, amb els paràmetres de posició, inclinació i escalat que en conjunt defineixen una transformació lineal). Com que això li permet calcular les coordenades ***SCU*** dels punts significatius de la inserció a partir de les coordenades ***SCO*** dels seus homòlegs en el bloc, AutoCAD controla la geometria del text: directament, les posicions transformades del punt d'ancoratge i del punt inicial; i a partir d'aquest, la del punt base.

Però si explosionem la inserció es trenquen els lligams amb la definició del bloc, i on hi havia un text deformat (un text geomètricament transformat per inserció del bloc del qual formava part) només quedarà un text independent i normalitzat, amb unes característiques que haurien de justificar la geometria per sí mateixes: la inclinació del text, l'alçada dels caràcters i la seva obliqüitat, considerant que $\varphi \neq 0°$ és l'única manera de legitimar que els pals de les lletres no siguin perpendiculars a la direcció d'escriptura. Ara bé: a l'hora de situar el text com Déu mana (amb els punts base i d'ancoratge alineats perpendicularment a aquesta direcció, amb independència de l'obliqüitat dels caràcters), ¿es donarà per bo el punt base transformat i a partir d'ell es restituirà el d'ancoratge, o es donarà

per bo el punt d'ancoratge i a partir d'ell es restituirà el de base? Si recordem que el punt base no és una dada de primera mà, a diferència del punt d'ancoratge, entendrem que se segueixi el segon camí. De fet, la seqüència podria anar així: AutoCAD obté la posició del punt d'ancoratge (aplicant la transformació d'inserció a la que figura sota codi 11 en el text de la definició del bloc) i la guarda sota el codi 11; obtingudes les noves alçada (codi 40) i orientació del text (codi 50), calcula el punt base (el situa des del punt precedent, perpendicularment a aquesta última), i a partir d'aquí el punt inicial (reculant sobre la línia base en funció de la longitud del text i la justificació horitzontal) que guarda sota el codi 10.

Aquesta (l'homologació del text com a objecte autònom, "regularitzant" la posició del punt base –o del punt inicial, si es vol- respecte al punt d'ancoratge) és la causa del desplaçament, tal i com s'esquematitza en la Figura 9.3 on, a sota de sengles textos "H" amb justificació ***SIZ***, ***MI*** i ***II*** n'hem situat uns de desplaçats precisament perquè els punts d'ancoratge no s'havien pres a partir del punt base transformat sinó just a l'inrevés (el punt base s'havia pres a partir dels punts d'ancoratge transformats).

Queda per argumentar per què, quan els components del bloc són textos o atributs C inclinats, cal explosionar la inserció perquè el fenòmen es manifesti, mentre que quan són atributs P o N inclinats, el desplaçament es dóna només inserir el bloc, referint-nos sempre a insercions en què $\mathbf{E_x \neq E_y}$:

- Els components del primer grup ja hem vist que, com a simples subjectes passius de la transformació d'inserció, en tenien prou amb una informació encreuada: la descripció dels objectes constitutius del bloc (podeu accedir al primer d'ells fent **(cdr (assoc -2 (tblsearch "BLOCK"** <nom_bloc>**)))** i aplicant reiteradament **entnext** fins arribar als components **"TEXT"** o **"ATTDEF"** desitjats, o accedir-hi directament seleccionant-los en la inserció amb **(car (nentsel))**, veient-ne el contingut amb **entget** quan els tingueu en el punt de mira), i els paràmetres de la inserció (podeu trobar-los en l'objecte **"INSERT"**, localitzat amb **entlast** o **(car (entsel))**, o en forma de matriu de transformació, amb **(caddr (nentsel))**). En explosionar la inserció, l'emancipació del text o de l'atribut C obligava a substituir aquestes fonts per informació geomètrica concreta, i la restitució d'algunes dades en base a uns supòsits falsos provocava l'error (per copsar la diferència amb la informació precedent i la similitud amb la relativa a atributs P i N, que veurem tot seguit, podeu veure la descripció dels subproductes de l'explosió fent **(entget (car (entsel)))**).
- Els del segon grup, malgrat tenir la majoria de característiques predeterminades en el bloc, no serà fins inserir-lo o, després d'això, editar-lo (procés que pot afectar no només el valor sinó la geometria) que adoptarà una forma definitiva. Per això AutoCAD destina a cada atribut P o N un paquet d'informació addicional (objecte **"ATTRIB"**, que cal no confondre amb l'objecte **"ATTDEF"** constitutiu del bloc), annex a l'objecte **"INSERT"**, que en estructura i contingut (la informació geomètrica ha de ser autosuficient, i respondre a les posicions transformades -eventualment modificades amb ***–ATREDIT***–) serà pràcticament idèntic al del text resultant d'explosionar un bloc del primer grup, incloent-hi el fet que el punt base es restitueix a partir del punt d'ancoratge transformat: darrera l'objecte **"INSERT"** segueixen aquests **"ATTRIB"**, rubricats pel senyal **"SEQEND"**; els podeu veure seleccionant la inserció amb l'ordre ***LIST***, però també localitzant-la amb **entlast** o **(car (entsel))** i aplicant reiteradament **entnext**, o seleccionant cada atribut amb **(car (nentsel))** per accedir-hi directament (una vegada a l'adreça correcta, **entget** permet veure'n el contingut). La prova que l'objecte **"ATTRIB"** és ja un resultat final i no una primitiva a transformar, és que la llista que s'obté de la funció **nentsel** no inclou cap referència a la transformació (aquí **(caddr (nentsel))** ja no existeix, a diferència del que passava seleccionant un component **"TEXT"** o **"ATTDEF"** del primer grup).

En resum: la geometria dels textos i dels atributs C en cada inserció d'un bloc no figura explícitament a la base de dades, i cal explosionar les insercions perquè quedi registrada com la de qualsevol objecte independent (tot i que en el cas dels atributs C això comportarà retrocedir a l'estatus de preassignació); per contra, la dels atributs P i N queda registrada de bon principi en totes les insercions (altrament, no serien editables). El desplaçament de textos i atributs es produeix per un error en la determinació d'aquesta geometria, degut a la incompatibilitat entre dues decisions: assimilar la deformació angular a un canvi en l'obliqüitat dels caràcters, pressuposant així que els punts base i d'ancoratge se situaran en una mateixa perpendicular a la línia base, i calcular el punt base a partir de la posició transformada del punt d'ancoratge, en comptes de fer-ho a l'inrevés.

Probablement ens haguem estès massa tractant de trobar una explicació plausible a aquests desplaçaments i ja sigui hora d'ocupar-nos de la seva quantificació, però abans volem presentar una altra incidència, que també afecta textos i atributs en blocs inserits amb factors d'escala $\mathbf{E_x \neq E_y}$ i que haurem d'afrontar tard o d'hora, tot i no interferir en el disseny de **C:INSERTOK**. Preferim fer-ho ara perquè així ho despatxarem de forma més expeditiva: malgrat produir-se en unes circumstàncies també ben delimitades i ser de fàcil quantificació, ens trobem davant d'un error sense solta ni volta, que ni tan sols sembla obeir a una lògica defectuosa (com en el cas dels desplaçaments), raó per la qual no perdrem el temps en especulacions. Anant al gra,

En quines circumstàncies es produeix el fenòmen?

Només s'observa en textos i atributs definits amb caràcters oblics (1ª diferència amb els desplaçaments, que es produïen amb independència de l'obliqüitat) i cal explosionar una inserció (2ª diferència, ja que els atributs P i N es desplaçaven sense descompondre-la) en què $\mathbf{E_x \neq E_y}$. En canvi, l'orientació de textos i atributs en el bloc és irrellevant (3ª diferència), perquè fins i tot quan són horitzontals resulten afectats.

En què consisteix?

En l'aixafament dels caràcters, l'alçada dels quals minva en la proporció $\cos\varphi$ (φ és l'angle d'obliqüitat del text o atribut en la definició del bloc), sense cap variació en l'amplada.

De fet, podríem afirmar que l'afectació pel coeficient $\cos\varphi$ és universal però que quan els caràcters són rectes ($\varphi = 0°$) no hi ha minoració. Però allò que realment desconcerta és el comportament quan el bloc ha estat definit amb caràcters oblics: si s'insereix amb escalat uniforme no hi ha escurçament, però així que incrementem un dels factors d'escala (mantenint l'angle de gir, per no introduir en el joc una nova variable) passarem a l'afectació assenyalada, que es mantindrà constant en el valor $\mathbf{h}\cos\varphi$, encara que seguim incrementant aquest factor. Si l'alçada hagués anat evolucionant de forma continua, responent per exemple a una funció del tipus $\mathbf{k}\,\frac{\mathbf{E_x}}{\mathbf{E_y}}\,\mathbf{h}\cos\varphi$, si més no tindríem un punt de partença per aventurar interpretacions com hem fet amb els desplaçaments, però la discontinuïtat descrita, juntament amb l'absurd que el coeficient minorador sigui precisament $\cos\varphi$ (com si, en comptes de prendre com alçada la distància recta entre les línies base i superior, en la definició del bloc, l'hagués pres en la direcció oblíqua dels caràcters), sols ens omple de perplexitat, i com que en el nostre objectiu de dissenyar eines eficients només estem obligats a detectar les fallades d'AutoCAD per poder-les contrarestar, i no pas jugar a detectius esbrinant què podia dur els seus creadors a espifiar-la de tant en tant, donarem el tema per tancat.

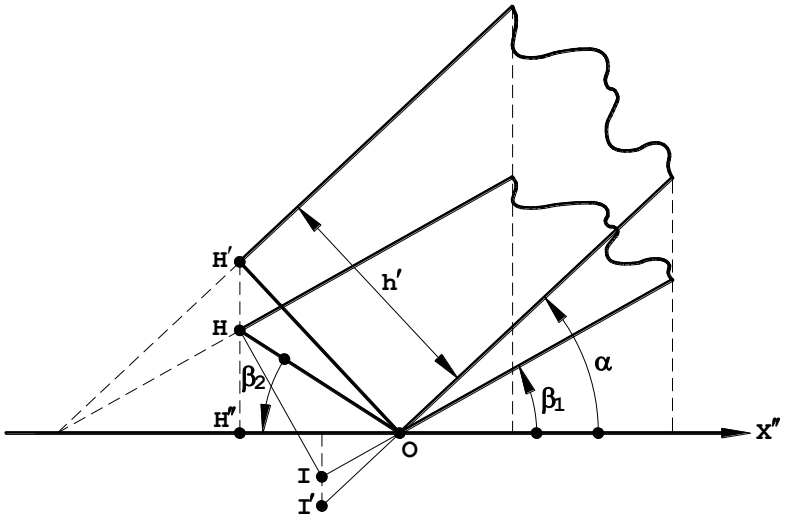
Acabada la digressió a propòsit de l'aixafament de textos i atributs constitutius de blocs, tornem a la questió del seu desplaçament sobre la direcció d'escriptura, descrit geomètricament però del qual en quedava pendent la quantificació numèrica.

Igual que havíem fet en l'anàl·lisi geomètrica del capítol precedent, considerarem la inserció de **BLOC*** amb factors d'escala $\mathbf{E_x}$ i $\mathbf{E_y}$ descomposta en dues: inserció de **BLOC***, girada i amb escalat uniforme $\mathbf{E_{1x} = E_{1y} = E_x}$, seguida d'una inserció de la precedent (prèviament convertida en bloc) sense girar, amb el punt d'inserció desplaçat de tal manera que els punts base de l'atribut considerat coincideixin (en rigor, hauríem de parlar de coincidència entre el punt base de l'atribut en la primera inserció i el seu homòleg en la segona, després de les puntualitzacions que hem hagut de fer) i amb factors d'escala $\mathbf{E_{2x} = 1}$ i $\mathbf{E_{2y}} = \frac{\mathbf{E_y}}{\mathbf{E_x}}$ (òbviament, serà $\mathbf{E_x = E_{1x}\,E_{2x}}$ i $\mathbf{E_y = E_{1y}\,E_{2y}}$). En la Figura 9.4 representem les dues insercions, on $\mathbf{h'}$ seria el producte de $\mathbf{E_x}$ per la distància de la línia d'on pengem l'atribut a la línia base (si és la línia superior, aquesta distància coincidirà amb l'alçada dels caràcters; si és la línia mitjana, farà la meitat; si és la línia inferior, farà un terç), de manera que el punt d'ancoratge, que haurà quedat a $\mathbf{H'}$ després de la primera inserció, se situaria a **H** amb la segona, que podríem considerar una transformació d'homologia afí respecte l'eix **O–X″** (paral·lel a l'eix **X** del bloc, i determinat pel doble punt base i l'angle de gir), en ser **H–H′** perpendicular a

aquest eix. Anàlogament, el vector **O–I** de la figura seria el producte de $\mathbf{E_x}$ pel desplaçament real de l'atribut, així que:

$$\text{desplaçament} = \frac{\mathbf{O–I'}}{\mathbf{E_x}} = \frac{\mathbf{O–H}\ \cos\left(\beta_1 + \beta_2\right)}{\mathbf{E_x}} = \frac{1}{\mathbf{E_x}}\sqrt{\mathbf{O–H''}^2 + \mathbf{H–H''}^2}\ \cos\left(\beta_1 + \beta_2\right)$$

Figura 9.4



Anem ara a posar tots els termes en funció de **h**, α, $\mathbf{E_x}$ i $\mathbf{E_y}$:

$$\mathbf{O–H''} = \mathbf{h'}\ \cos(90° - \alpha) = \mathbf{h'}\ \sin\alpha$$

$$\mathbf{H–H''} = \mathbf{E_{2y}}\ \mathbf{H'–H''} = \mathbf{E_{2y}}\ \mathbf{h'}\ \sin\left(90° - \alpha\right) = \mathbf{E_{2y}}\ \mathbf{h'}\ \cos\alpha = \frac{\mathbf{E_y}}{\mathbf{E_x}}\ \mathbf{h'}\ \cos\alpha$$

$$\beta_1 = \arctan\left(\mathbf{E_{2y}}\ \tan\alpha\right) = \arctan\left(\frac{\mathbf{E_y}}{\mathbf{E_x}}\ \tan\alpha\right)$$

$$\beta_2 = \arctan\left(\mathbf{E_{2y}}\ \tan\left(90° - \alpha\right)\right) = \arctan\ \frac{\mathbf{E_{2y}}}{\tan\alpha} = \arctan\ \frac{\mathbf{E_y}}{\mathbf{E_x}\ \tan\alpha}$$

Si el que esteu pensant és que no en tenim prou a conèixer el desplaçament $\frac{\mathbf{O–I}}{\mathbf{E_x}}$ de cada atribut N de **BLOC*** en la inserció final d'aquest genèric, sinó que convindria saber a quin desplaçament virtual $\frac{\mathbf{O–I'}}{\mathbf{E_x}}$ (en la inserció explosionada de **BLOC-2**) es correspon l'anterior, per avançar-nos a l'efecte no desitjat, contrarestant-lo *a priori* (resituant l'atribut amb ***DESPLAZA*** abans de crear **BLOC***, amb un desplaçament de signe contrari, per tal que, en inserir el genèric, es desvii precisament fins allà on el volíem), us hem de dir que una estratègia així no ens portaria enlloc. De fet, pàgines enrera, quan preparàvem una VERSIÓ 10 que ens havia de permetre corregir la inversió de l'obliqüitat dels atributs en determinades circumstàncies (abans que anéssim a l'arrel del problema, assegurant-nos que tots els **BLOC***s es definien en base a insercions de **BLOC** fetes amb $\mathbf{E_y > 0}$), ja us preveníem contra aquesta falsa solució, dient que només aconseguiríeu que el procediment funcionés en la inserció inaugural de **BLOC*** i en totes aquelles en què la relació de factors d'escala $\frac{\mathbf{E_x}}{\mathbf{E_y}}$ tingués el mateix valor (insercions **BLOC*** geomètricament semblants), però per a valors diferents, la distorsió prèvia seria insuficient o excessiva.

Igual que allà, si no considerem la possibilitat d'evitar les condicions en què es produeix l'anomalia, no quedarà més sortida que esmenar-la *a posteriori*, inserció a inserció. Però també com allà, quan ja disposem d'una funció **CORR-POSICIO** capaç de corregir la posició de cada atribut N desplaçant-lo el vector **I–O**, el mòdul del qual calcularà segons l'expressió obtinguda línies enrera,

```
(defun CORR-POSICIO (/ H A SINA COSA TANA OI)
  (setq H (* (cdr (assoc 40 LE*))
             (if (= C-72 4)
                 0.5
                 (if (= C-74 1)
                     (/ 1.0 -3)
                     (if (= C-74 2) 0.5 1))))
        A (cdr (assoc 50 LE*)) SINA (sin A) COSA (cos A)
        TANA (if (not (equal COSA 0 Q0)) (abs (/ SINA COSA)))
        OI (if TANA (* H (sqrt (+ (expt (* EX SINA) 2) (expt (* EY COSA) 2)))
                       (cos (+ (atan (* EY/EX TANA)) (atan (/ EY/EX TANA)))))
                    0))
  (polar (cdr (assoc 11 LE)) (cdr (assoc 50 LE)) OI))
```

tant si recorrem a l'ordre ***-ATREDIT*** (com hem assenyalat en el penúltim capítol, en relació a **C:RATREDIT**, si l'ordre és executada des de **command** només accepta editar un atribut per inserció, així que haurem d'iterar-la tantes vegades com atributs N tingui **BLOC***) i el codi afegit a **INSERT*** és

```
(if OO-2
    (progn
      (setq EX OX** EY (distance Y** Y) EY/EX (/ EY EX)
            E (entlast) E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))
      (PASSA-ATTRIB)
      (while (setq E (entnext E) E* (entnext E*))
             (if (or (= (setq LE (entget E) LE* (entget E*)
                              C-72 (cdr (assoc 72 LE))) 4)
                         (and (< C-72 3) (> (setq C-74 (cdr (assoc 74 LE))) 0)))
                 (command "ATREDIT" "" "" "" ""
                          E "P" (trans (CORR-POSICIO) E 1) ""))))))
```

com si ens servim de les funcions **entmod** i **entupd**, amb l'afegitó

```
(if OO-2
    (progn
      (setq EX OX** EY (distance Y** Y) EY/EX (/ EY EX)
            E (entlast) E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))
      (PASSA-ATTRIB)
      (while (setq E (entnext E) E* (entnext E*))
             (if (or (= (setq LE (entget E)
                              LE* (entget E*)
                              C-72 (cdr (assoc 72 LE))) 4)
                         (and (< C-72 3) (> (setq C-74 (cdr (assoc 74 LE))) 0)))
                 (entmod (subst (cons 11 (CORR-POSICIO)) (assoc 11 LE) LE))))
      (entupd (entlast)))))
```

caldrà desempolsar la funció **VAL-ATRIBS** (que no utilitzàvem des de la VERSIÓ 7), per generar la llista **WWAA** de valors d'atributs N i aconseguir que la inserció de **BLOC*** es tanqui amb aquests valors. Si no, en el primer cas tindríem el conflicte servit (entre l'expressió **(command "CLAYER" CAPA "INSERT" BLOC* ...)** que, en no disposar d'arguments que aportessin valors per als atributs de **BLOC***, els aniria a buscar entre els de les reiterades **(command "ATREDIT" ...)**) i en el segon cas la intervenció de **entmod** seria prematura (no trobaria cap atribut per esmenar-ne la posició, perquè **(entnext (entlast))** encara seria **nil**).

A més, com que l'obtenció de les dades necessàries per al càlcul de la correcció implica rastrejar simultàniament en dues fonts diferents (els objectes **"ATTRIB"** de la inserció de **BLOC***, d'on treurem els valors associats als codis 11 i 50, per al punt d'ancoratge i l'angle de inclinació en el dibuix, i els objectes **"ATTDEF"** de la definició de **BLOC***, d'on treurem els valors associats als codis 40 i 50, per a l'alçada dels caràcters i l'orientació de l'atribut en el bloc, a més dels codis 72 i 74 per a la justificació horitzontal i vertical, que podem treure de una o altra font), cal assegurar-se que els elements **E** i **E*** d'ambdues sèries de dades sempre corresponen a un mateix atribut N. I, com que la presència d'atributs P pot complicar les coses (en la definició de **BLOC*** figuraran com a objectes **"ATTRIB"** de la inserció de **BLOC-1**, rematats per l'objecte-senyal **"SEQEND"**, i s'interposaran entre aquesta inserció i els atributs N que ens interessen), hem decidit muntar el dispositiu **PASSA-ATTRIB** per saltar-nos-els, al qual hem donat forma de funció per utilitzar-lo més endavant. Com que tot s'ha anat complicant i l'ús de **WWAA** porta cua (retornar **command** al format de llista sense avaluar i recuperar l'altra vegada operativa assignació **(setvar "INSNAME" BLOC)**), reproduïm la versió sencera, en què ens hem decantat per l'ús de la funció **entmod**, més ràpida que l'ordre ***-ATREDIT***:

```
; VERSIÓ 14+

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun MAKEBLOC (BLOC/2 OO ATRIBS)
  (entmake (list '(0 . "BLOCK")
                 (cons 2 BLOC/2)
                 '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun REDEF-BLOC*S (/ BLOK*)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (setvar "ATTREQ" 0)
  (while LB
         (if (wcmatch (setq BLOK* (cdr (assoc 2 LB))) N)
             (progn
               (setq E (cdr (assoc -2 LB))
                     LE (entget E)
                     J (cdr (assoc 41 LE))
                     K (/ (* (cdr (assoc 50 LE)) 180) PI))
               (command "INSERT" BLOC-1 '(0 0 0) J J K)
               (setq SS (ssadd (entlast)))
               (if OO-2
                   (progn
                     (command "INSERT" BLOC-2 '(0 0 0) J J K
                              "DESCOMP" (entlast))
                     (setq SA (ssget "P") K -1)
                     (while (setq E (ssname SA (setq K (1+ K))))
                            (ssadd E SS))))
               (command "BLOQUE" BLOK* "S" '(0 0 0) SS ""))
         (setq LB (tblnext "BLOCK")))
  (setvar "ATTREQ" (if ATREQ 1 0))
  (setvar "CLAYER" CAPA))

(defun SEGR-ATRIBS (/ BL1EX BL2EX AT AT-CP LB)
  (setq BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2))
  (if (and BL1EX BL2EX)
      (setq OO-2 (/= (cdr (assoc 0 (entget (cdr (assoc -2 BL2EX))))) "ENDBLK"))
```

```
      (progn
        (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (cdr (assoc 70 LE)) 10) 0))
                   (setq OO-2 (cons LE OO-2))
                   (setq OO-1 (cons LE OO-1)
                         AT-CP (if AT-CP T AT)))
               (setq E (entnext E)))
        (tblnext "BLOCK" T)
        (while (and (setq LB (tblnext "BLOCK"))
                    (not (setq BL*EX (wcmatch (cdr (assoc 2 LB)) N)))))
        (setvar "CECOLOR" "PORCAPA")
        (setvar "CELTYPE" "PORCAPA")
        (setvar "CELWEIGHT" -1)
        (MAKEBLOC BLOC-1 OO-1 AT-CP)
        (MAKEBLOC BLOC-2 OO-2 OO-2)
        (if BL*EX (REDEF-BLOC*S))
        (setvar "CECOLOR" COL)
        (setvar "CELTYPE" TLIN)
        (setvar "CELWEIGHT" GLIN))))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS (/ ICVP M VD V LLAA)
  (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC-2))))
  (while E
         (setq LE (entget E)
               ICVP (cdr (assoc 70 LE)))
         (if (and (not LLAA) ATREQ) (prompt "\nIndique valores de atributo"))
         (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LE))
               (cdr (assoc 3 LE)) (cdr (assoc 1 LE)))
         (setq LLAA (cons (list W N M V) LLAA)
               E (entnext E)))
  (setq W ())
  (foreach LA (reverse LLAA)
    (if (car LA)
        (progn
          (if (and (not W) ATREQ) (prompt "\nVerificar valores de atributo"))
          (VATR (cons "" W) (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq WWAA (append WWAA (list V))))
  (setq WWAA (append WWAA W)))

(defun PASSA-ATTRIB ()
  (while (wcmatch (cdr (assoc 0 (entget (entnext E*)))) "ATTRIB,SEQEND")
         (setq E* (entnext E*))))

(defun CORR-POSICIO (/ H A SINA COSA TANA OI)
  (setq H (* (cdr (assoc 40 LE*))
             (if (= C-72 4)
                 0.5
                 (if (= C-74 1)
                     (/ 1.0 -3)
                     (if (= C-74 2) 0.5 1))))
        A (cdr (assoc 50 LE*)) SINA (sin A) COSA (cos A)
        TANA (if (not (equal COSA 0 Q0)) (abs (/ SINA COSA)))
        OI (if TANA (* H (sqrt (+ (expt (* EX SINA) 2) (expt (* EY COSA) 2)))
                       (cos (+ (atan (* EY/EX TANA)) (atan (/ EY/EX TANA)))))
                    0))
  (polar (cdr (assoc 11 LE)) (cdr (assoc 50 LE)) OI))
```

```
(defun INSERT* (/ BLOC-1 BLOC-2 BLOC* BL*EX WWAA OO-1 OO-2 E E* LE LE* SS SA
                  X* X** Y** OX* OX** C-72 C-74 EX EY EY/EX)
  (setq W (if (< J K) (angle B X) (angle X B))
        B (mapcar '/ (mapcar '+ B X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*)
        OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
  (SEGR-ATRIBS)
  (if OO-2 (VAL-ATRIBS))
  (if (not (setq BL*EX (tblsearch "BLOCK" BLOC*)))
      (progn
        (command "CLAYER" "0"
                 "CECOLOR" "PORBLOQUE"
                 "CELTYPE" "PORBLOQUE"
                 "CELWEIGHT" -2
                 "INSERT" BLOC-1 O (setq K (/ OX* OX**)) K X*)
        (setq SS (ssadd (entlast)))
        (if OO-2
            (progn
              (command "ATTREQ" 0
                       "INSERT" BLOC-2 O K K X*
                       "ATTREQ" (if ATREQ 1 0)
                       "DESCOMP" (entlast))
              (setq SA (ssget "P") K -1)
              (while (setq E (ssname SA (setq K (1+ K))))
                     (ssadd E SS))))
        (command "SCP" "Z" (setq K (trans O 1 0) O O) X**
                 "BLOQUE" BLOC* (trans K 0 1) SS ""
                 "SCP" "PR"
                 "CELWEIGHT" GLIN
                 "CELTYPE" TLIN
                 "CECOLOR" COL
                 "CLAYER" CAPA)))
  (eval (append '(command "INSERT" BLOC* O OX** (* (distance Y** Y) I) X**)
                (if ATREQ WWAA)))
  (if OO-2
      (progn
        (setq EX OX** EY (distance Y** Y) EY/EX (/ EY EX)
              E (entlast) E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))
        (PASSA-ATTRIB)
        (while (setq E (entnext E) E* (entnext E*))
               (if (or (= (setq LE (entget E)
                                LE* (entget E*)
                                C-72 (cdr (assoc 72 LE))) 4)
                       (and (< C-72 3) (> (setq C-74 (cdr (assoc 74 LE))) 0)))
;                   Les 2 línies que segueixen són una versió prèvia i més lenta.
;                   (command "ATREDIT" "" "" "" ""
;                            E "P" (trans (CORR-POSICIO) E 1) "")))))
                   (entmod (subst (cons 11 (CORR-POSICIO)) (assoc 11 LE) LE))))
        (entupd (entlast)))))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC N O X Y
                     OX OY XY A B W I J K)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (setq PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
```

```
      ATREQ (= (getvar "ATTREQ") 1)
      O (getvar "INSNAME")
      BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
      BLOC (if (= BLOC "")
               (if (> O "") O (prompt "\nNombre de bloque no válido."))
               (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                            (< (cdr (assoc 70 O)) 4))
                       (and (not O) (findfile (strcat BLOC ".dwg"))))
                   BLOC (alert (if O (REFX) (RUTES)))))
      BLOC (if BLOC (strcase BLOC) (exit))
      N (strcat BLOC "_" N)
      O (getpoint "\nPrecise punto de inserción: "))
(foreach P '("X" "Y")
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                  " desde el punto de inserción: "))
            I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC " <1>: "))
            W (angle O A)
            B (/ (distance O A) (if I I 1)))
      (set (read P) (polar O W B))
      (set (read (strcat "O" P)) B))
(setq A (polar O (+ W PI/2) OY)
      B (polar O (- W PI/2) OY)
      XY (distance X Y)
      I (if (< (distance A X) (distance B X)) -1 1))
(setvar "CMDECHO" 0)
(setvar "OSMODE" 0)
(setvar "ATTDIA" 0)
(if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
    (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
        (progn
          (command "INSERT" BLOC O OX (* OY I) X)
          (prompt (if (> (getvar "CMDACTIVE") 0)
                      "\nIndique valores de atributo\n" "\n")))
        (INSERT*)))
(setvar "INSNAME" BLOC)
(setvar "ATTDIA" ATDIA)
(setvar "OSMODE" OSN)
(setvar "CMDECHO" ECO)
(princ))
```

Tot plegat, l'arranjament d'aquesta última deficiència ens ha sortit prou car: la reintroducció de **VAL-ATRIBS** per formar la llista **WWAA** de valors d'atribut (amb la impertinent reaparició del missatge *Verificar valores de atributos*, repetit fora de lloc quan tenim atributs N amb aquesta característica) i el laboriós processat de **CORR-POSICIO**; això en cada execució no trivial de **C:INSERTOK**, i no només en les insercions inaugurals de cada genèric **BLOC*.** Però si considerem que la feina no es pot donar per acabada amb **C:INSERTOK**, perquè caldria posar a l'abast de l'usuari una eina (una altra funció, diguem-ne **C:DESCOMPOK**) que permetés aplicar ***DESCOMP*** a les insercions resultants de la primera, neutralitzant els "efectes col·laterals" que l'ús directe d'aquesta ordre comportaria, el balanç encara és pitjor, perquè, on teníem un problema a resoldre, la VERSIÓ 14+ de **C:INSERTOK** n'afegirà un altre. D'una banda, l'aixafament dels atributs N definits amb caràcters oblics, que fins ara no ens havia afectat, però que ho farà tan bon punt explosionem algun **BLOC*.** De l'altra, la manera en què la descomposició d'un bloc afecta els seus atributs: pèrdua de valors i retorn a l'estat de preassignació (cal suposar que l'usuari ja hi compta, amb això), és a dir, a les condicions de la definició de cada atribut, modificades pels factors d'escala i l'angle de gir de la inserció, però renunciant als canvis posteriors (amb això segur que l'usuari no hi compta, perquè ignora que el treball de ***INSERTOK*** es deu en bona part a la intervenció de ***-ATREDIT*** o **entmod**).

Així doncs, la nova ordre ***DESCOMPOK*** no només s'haurà d'ocupar de la minva d'alçada experimentada pels caràcters oblics, sinó de repetir part de l'accio de **C:INSERTOK** (resituar els atributs desplaçats, executant de nou **CORR-POSICIO**), que se n'haurà anat en orris en descompondre la inserció. Establert això, abans d'oferir el codi de **C:DESCOMPOK** ens permetrem uns aclariments:

- No farem res per suprimir o ocultar el missatge *<N> no se ha podido descomponer*, perquè correspon a ***DESCOMP*** i perquè no denota cap error d'execució: l'únic que

pretén dir-nos l'Editor de Dibuix (el missatge no és cap prodigi de claredat, sense ficar-nos en qüestions de concordança de nombre*) és que no podem tractar el primer component de **BLOC*** com la resta, deixant-lo com una inserció ordinària de **BLOC-1**, i mantenir alhora la forma geomètrica (adaptació a un paral·lelogram d'encaix). Com havíem comentat en relació a la frustrada VERSIÓ 5, el missatge s'ha d'interpretar en el sentit que la inserció de **BLOC-1** es transforma en la d'un bloc anònim (nom del tipus ***E**<**n**>) o ni això, si **BLOC*** no tenia atributs.

- L'excepcionalitat d'aquesta transformació fa que, el que inicialment era primer component de **BLOC***, constitueixi l'excepció a la regla (els objectes resultants d'explosionar un bloc s'arrengleren mantenint el mateix ordre que en el bloc) i passi a ser l'últim després de la descomposició. I, com que en cada atribut ens interessa disposar de dades relatives a la inserció de **BLOC*** (objecte **"ATTRIB"**, amb el codi 11 per al punt d'ancoratge, el codi 40 per a l'alçada efectiva dels caràcters, el codi 41 per al factor d'amplada i el codi 50 per a la orientació en el dibuix) i a la seva definició com a bloc (objecte **"ATTDEF"**, amb el codi 40 per a l'alçada de caràcters, el codi 50 per a l'angle d'inclinació de l'atribut en el bloc i el codi 51 per a l'angle d'inclinació dels caràcters respecte a la normal), dades que en algun cas és indiferent treure d'un lloc o l'altre (codis 72 i 74 per a la justificació horitzontal i vertical), per tal de garantir la correspondència entre els dos conjunts ordenats començarem agafant l'element **(ssname (ssget "P") 0)** de l'un (el primer del conjunt de selecció resultant de l'explosió, que seria el segon si no fos perquè el primer se n'ha anat a la cua) i l'element **(entnext (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))** de l'altre, que com a segon component serà el primer atribut N de la definició de **BLOC***, sempre que **BLOC** no tingui atributs P. Si en té, ja sabem que la funció **PASSA-ATTRIB** ens ajudarà a saltar-nos els **"ATTRIB"** de la inserció de **BLOC-1** fins accedir a aquest primer atribut N que, en haver tornat a l'estat de preassignació, és **"ATTDEF"**.
- Pel que fa a l'estirament vertical dels caràcters, en els atributs definits amb un estil de text oblic, i anomenant **E** els elements del primer conjunt i **E*** els del segon, cal contrarestar l'aixafament que havíem descrit, augmentant l'alçada i disminuint el factor d'amplada en la mateixa proporció, per deixar intacta l'amplada: dividirem **(cdr (assoc 40 LE))** i multiplicarem **(cdr (assoc 41 LE))** pel mateix valor **(cos (cdr (assoc 51 LE*)))**.

```
; continuació de VERSIÓ 14+

(defun C:DESCOMPOK (/ Q0 ECO BLOC* E E* LE LE* SS EX EY EY/EX A N)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione una inserción INSERTOK con atributos no justificados ")
  (prompt "sobre la línea")
  (prompt "\nbase o de caracteres oblicuos (usando DESCOMP se desplazan o ")
  (prompt "mengua su altura)")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              EX (cdr (assoc 41 LE))
              EY (cdr (assoc 42 LE))
              EY/EX (/ EY EX)
              ECO (getvar "CMDECHO"))
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (setvar "CMDECHO" ECO)
        (if (and (wcmatch BLOC* (strcat "*_" N)) (setq SS (ssget "P")))
            (progn
              (setq E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))) N -1)
              (PASSA-ATTRIB)
              (while (setq N (1+ N) E (ssname SS N)
                           E* (entnext E*))
                     (setq LE (entget E) LE* (entget E*)
                           A (cos (cdr (assoc 51 LE*))))
```

* En relació al nombre <*N*> del críptic missatge, que serà més gran que la unitat quan **BLOC** contingui atributs P, el lector d'aquestes línies n'haurà copsat el significat, però és més dubtós que l'usuari normalet arribi tan lluny: efectivament, AutoCAD no només compta l'objecte **"INSERT"** corresponent a la inserció de **BLOC-1** sinó els **"ATTRIB"** corresponents a aquests atributs P, a més del senyal **"SEQEND"**.

```
                    (if (not (equal A 1 Q0))
                        (progn
                          (setq LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A))
                                          (assoc 40 LE) LE)
                                LE (subst (cons 41 (* (cdr (assoc 41 LE)) A))
                                          (assoc 41 LE) LE))
                          (entmod LE)))
                    (if (or (= (setq C-72 (cdr (assoc 72 LE))) 4)
                            (and (< C-72 3)
                                 (> (setq C-74 (cdr (assoc 74 LE))) 0)))
                        (entmod (subst (cons 11 (CORR-POSICIO))
                                       (assoc 11 LE) LE)))))))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

Veient que aquests entremeliats atributs porten cua, venen ganes de fer-se enrera, renunciar a la VERSIÓ 14+ i anunciar: *Senyores i senyors: l'ordre* ***INSERTOK*** *només és aplicable a blocs amb atributs si aquests estan ubicats en relació a la línea base (punt inicial,* ***C****entre ,* ***D****reta, a****L****inear o a****J****ustar); si ho estan en relació a la línia superior (****SIZ****,* ***SC****,* ***SD****), mitjana (****MI****,* ***M*** *o* ***MC****,* ***MD****) o inferior (****II****,* ***IC****,* ***ID****), caldrà redefinir-los de manera que, mantenint l'aspecte, s'acomodin a l'imperatiu enunciat*. D'altra banda, com que sap greu començar a aplicar restriccions d'ús al producte, ens poden venir ganes d'atemptar contra els drets de l'usuari (això sí, disfressant l'operació de paternalisme i silenciant la mala consciència amb aquell argument, tan suat com fals, que en el fons ho fem pel seu bé): *¿per què molestar l'usuari, si la mateixa aplicació podria encarregar-se de redefinir el bloc, sense aixecar la llebre?*.

Refusem, però, aquests cants de sirena: quan l'usuari ha decidit ubicar d'aquesta manera els atributs de **BLOC**, les seves raons haurà tingut per fer-ho, i canviar-li això sense previ advertiment podria causar-li problemes fora de **C:INSERTOK**. A més, tard o d'hora acabarem adonant-nos que no hi havia cap necessitat de tocar **BLOC**, disposant dels substituts **BLOC-1** i **BLOC-2**, que són els qui al final donen la cara. Efectivament, un avantatge addicional de les versions 8+ i 13+ (la VERSIÓ 12+ en queda exclosa, perquè els blocs constituïts exclusivament per atributs N passen de blocs auxiliars) és que la subreptícia suplantació d'uns atributs per uns altres, d'igual aparença però ancorats a la línia base, pot quedar restringida a l'àmbit de **C:INSERTOK**: n'hi haurà prou a actuar sobre **BLOC-2**, sense afectar **BLOC**. Només caldrà canviar el punt d'ancoratge de cada atribut per la seva projecció sobre la línia base (canviant les coordenades guardades sota el codi 11), i la modalitat de justificació per les corresponents a aquesta línia:

- Si aquesta modalitat és ***M****edio* (codi_72 = **4**; codi_74 irrellevant), de primer la convertirem a mode ***MC*** (codi_72 = **1**; codi_74 = **2**), de comportament prou semblant.
- Els modes ***SD***, ***MD*** i ***ID*** (codi_72 = **2**; codi_74 = **3**, **2** i **1**) els passarem a ***D****erecha* (deixant codi_72 = **2** i fent codi_74 = **0**).
- Els modes ***SC***, ***MC*** i ***IC*** (codi_72 = **1**; codi_74 = **3**, **2** i **1**) els passarem a ***C****entro* (deixant codi_72 = **1** i fent codi_74 = **0**).
- Els modes ***SIZ***, ***MI*** i ***II*** (codi_72 = **0**; codi_74 = **3**, **2** i **1**) els passarem a *punto inicial* (deixant codi_72 = **0** i fent codi_74 = **0**).

Muntarem aquesta conversió a cavall de la classificació dels components de **BLOC**, prèvia a la creació (o recreació) de **BLOC-1** i **BLOC-2**, a càrrec de **SEGR-ATRIBS**, amb l'ajut de la nova funció **LINIA-BASE**. Veiem-les totes dues, prenent la primera de la VERSIÓ 13+ (l'afectació seria igual en la VERSIÓ 8+):

```
(defun LINIA-BASE ()
  (polar (cdr (assoc 11 LE))
         (- (cdr (assoc 50 LE)) PI/2)
         (* (cdr (assoc 40 LE))
            (if (= C-74 1)
                (/ 1.0 -3)
                (if (= C-74 2) 0.5 1)))))

(defun SEGR-ATRIBS (/ BL1EX BL2EX AT AT-CP C-72 C-74 LB)
  (setq BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2))
```

```
(if (and BL1EX BL2EX)
    (setq OO-2 (/= (cdr (assoc 0 (entget (cdr (assoc -2 BL2EX))))) "ENDBLK"))
    (progn
      (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
      (while E
             (setq LE (entget E))
             (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                      (= (logand (cdr (assoc 70 LE)) 10) 0))
                 (setq C-72 (cdr (assoc 72 LE))
                       C-74 (cdr (assoc 74 LE))
                       LE (if (= C-72 4)
                              (subst (cons 72 (setq C-72 1))
                                     (cons 72 4)
                                     (subst (cons 74 (setq C-74 2))
                                            (assoc 74 LE)
                                            LE))
                              LE)
                       LE (if (and (< C-72 3) (> C-74 0))
                              (subst (cons 74 0)
                                     (assoc 74 LE)
                                     (subst (cons 11 (LINIA-BASE))
                                            (assoc 11 LE)
                                            LE))
                              LE)
                       OO-2 (cons LE OO-2))
                 (setq OO-1 (cons LE OO-1)
                       AT-CP (if AT-CP T AT)))
             (setq E (entnext E)))
      (tblnext "BLOCK" T)
      (while (and (setq LB (tblnext "BLOCK"))
                  (not (setq BL*EX (wcmatch (cdr (assoc 2 LB)) N)))))
      (setvar "CECOLOR" "PORCAPA")
      (setvar "CELTYPE" "PORCAPA")
      (setvar "CELWEIGHT" -1)
      (MAKEBLOC BLOC-1 OO-1 AT-CP)
      (MAKEBLOC BLOC-2 OO-2 OO-2)
      (if BL*EX (REDEF-BLOC*S))
      (setvar "CECOLOR" COL)
      (setvar "CELTYPE" TLIN)
      (setvar "CELWEIGHT" GLIN))))
```

Tot i que aquesta solució ja està prou bé per la seva simplicitat de codi, encara podríem filtrar una mica més l'accés a la funció **LINIA-BASE**. De fet, en l'últim cas (modes ***SIZ***, ***MI*** i ***II***) no hauria calgut modificar el punt d'ancoratge (codi 11), perquè en la modalitat *punto inicial* és irrellevant: en els modes de justificació *a**L**inear* i *a**J**ustar* tant compta el punt inicial (codi 10) com el punt d'ancoratge (codi 11); en els modes ***SC***, ***MC*** i ***IC*** només compta el segon, tot i que el primer adopta el valor que li correspon (en funció de la longitud del nom identificador de l'atribut); però en el mode *punto inicial* només compta el primer (AutoCAD dóna per suposat que codi_11 = codi_10) i, com que en ***SIZ***, ***MI*** i ***II*** el punt inicial és la projecció del punt d'ancoratge sobre la línia base, el simple canvi a modalitat *punto inicial* ja comporta canviar de posició "el punt que compta". Considerant-ho així, l'afegitó precedent a **SEGR-ATRIBS** quedaria un pèl més llarg:

```
                    ................................................
                    (setq C-72 (cdr (assoc 72 LE))
                          C-74 (cdr (assoc 74 LE))
                          LE (if (= C-72 4)
                                 (subst (cons 72 (setq C-72 1))
                                        (cons 72 4)
                                        (subst (cons 74 (setq C-74 2))
                                               (assoc 74 LE)
                                               LE))
                                 LE)
                          LE (if (and (< C-72 3) (> C-74 0))
                                 (subst (cons 74 0)
                                        (assoc 74 LE)
                                        (if (> C-72 0)
                                            (subst (cons 11 (LINIA-BASE))
```

```
                                                (assoc 11 LE)
                                                LE)
                                          LE))
                                    LE)
                              OO-2 (cons LE OO-2))
                        ....................................................
```

Però no hem de caure en el parany de generalitzar el tracte dispensat als modes ***SIZ***, ***MI*** i ***II***, pensant que l'estratègia també és aplicable a la resta i que podem prescindir de la funció **LINIA-BASE**, substituint aquest fragment de **SEGR-ATRIBS** per

```
                        ....................................................
                        (setq C-72 (cdr (assoc 72 LE))
                              C-74 (cdr (assoc 74 LE))
                              LE (if (or (= C-72 4) (and (< C-72 3) (> C-74 0)))
                                     (subst (cons 72 0)
                                            (assoc 72 LE)
                                            (subst (cons 74 0)
                                                   (assoc 74 LE)
                                                   LE))
                                     LE)
                              OO-2 (cons LE OO-2))
                        ....................................................
```

perquè, tret dels modes esmentats (i de *punto inicial*, ***C****entro* i ***D****erecha*, és clar), això sols funcionaria quan el text assignat fos tan llarg com el nom de l'atribut.

Més enllà de **C:INSERTOK**, la regularització de **BLOC-2** quant a mode de justificació dels seus components ens permetrà de simplificar dràsticament **C:DESCOMPOK**, en la mesura que l'ordre ***DESCOMP*** ja no desbaratarà una acció correctora que en la VERSIÓ 14+ estava a càrrec de ***-ATREDIT*** o **entmod**, obligant-nos a aplicar-la per segon cop: només haurem d'estirar els caràcters dels atributs definits amb un estil de text oblic, augmentant l'alçada i disminuint el factor d'amplada una mateixa proporció, per deixar intacte l'amplada.

Els indubtables avantatges del sistema en relació a la VERSIÓ 14+ només tenen una contrapartida, que era perfectament previsible: si **BLOC** té atributs N no ubicats en relació a la línia base, quan vulguem editar-los en les insercions de **BLOC*** amb l'ordre ***-ATREDIT***, utilitzant determinades opcions es posarà de manifest que els punts d'ancoratge no són els mateixos que tenien a **BLOC**, tot i que això no és cap obstacle insalvable: sols comportarà la necessitat de desplaçaments addicionals per obtenir l'efecte desitjat. Així, editant-los un per un:

- ***P****osición*, si movem el cursor, revelarà que el punt d'ancoratge se situa sobre la línia base. Si, malgrat tot, donem una posició pensada per al punt d'ancoratge de **BLOC** (és a dir, per a les línies superior, mitjana o inferior), sempre queda el recurs d'usar ***P****osición* un altre cop per corregir la desviació, però serà millor realitzar alguns preparatius abans de l'edició, per tal d'encertar-la a la primera: per exemple, dibuixar una línia des del punt d'ancoratge virtual (el que correspon a la definició de **BLOC**) al punt d'ancoratge real en la inserció de **BLOC*** (el que correspon a **BLOC-2**) i desplaçar-la des del primer punt fins a la nova posició pensada per a ell; en executar ***-ATREDIT***, a requeriment de ***P****osición* no introduirem aquesta posició sinó l'altre extrem de la línia.
- ***A****ltura* i *áng****U****lo* tenen el cursor elàstic lligat al punt d'ancoratge real, fet que no influeix en la determinació numèrica o gràfica de la magnitud sol·licitada, però que ja dóna a entendre que aquest punt serà la base d'aplicació de l'alçada o el centre del gir. Com en el cas precedent, després podrem usar ***P****osición* per reubicar l'atribut, però abans d'embarcar-nos en ***-ATREDIT*** convindria realitzar alguns preparatius per tal de no actuar tan a cegues: per exemple, dibuixar dos punts-objecte sobre el punt d'ancoratge virtual, aplicar-li una transformació a un d'ells, amb el punt d'ancoratge real com a punt base (per a ***A****ltura*, ***ESCALA***, usant ***R****eferencia* i les altures actual i nova per determinar el factor d'escala; per a *áng****U****lo*, **GIRA**) i dibuixar una línia del punt transformat al no transformat, desplaçant-la des de l'extrem inicial al punt d'ancoratge real; quan executem ***-ATREDIT***, després de ***A****ltura* o de *áng****U****lo* anirem a ***P****osición* i introduirem l'altre extrem de la línia desplaçada.

(En alguns casos, la visualització esquemàtica dels atributs, proporcionada per l'ordre ***LOCTEXTO***, pot facilitar els preparatius recomanats, si bé convindrà que algun caràcter garanteixi la representació de la línia superior, saber si la línia oposada és la de base o la inferior, i tenir present que les altres dues només reflectiran la direcció ***SC-MC-C-IC*** si l'estil de text utilitzat no té obliqüitat.)

Parlant d'editar atributs, no estaria gens bé passar de puntetes sobre una qüestió que, si no s'esmenta, podria defraudar les expectatives de l'usuari i provocar-li algun ensurt, tot i ser de fàcil tractament: l'edició d'atributs P (predefinits). Com sabeu, aquests atributs són editables igual que els N (normals), però no heu d'oblidar que, de la VERSIÓ 8 ençà, els atributs P de cada genèric **BLOC*** no són components directes sinó que ho són d'una inserció de **BLOC-1**. I, si recordeu què passava en una de les primeres estacions del nostre particular viacrucis, en unes circumstàncies força semblants (a la VERSIÓ 4, cada **BLOC*** tenia una inserció de **BLOC** com a únic component, amb la particularitat que els atributs N hi figuraven amb uns mateixos valors provisionals que després havien de ser substituïts pels específics, amb ***-ATREDIT Valor***), copsareu la impossibilitat de reassignar valors diferents a atributs homònims de diferents insercions d'un mateix genèric **BLOC***: no pas perquè els atributs P se situïn a nivell 2 i no a 1 (que en això ***-ATREDIT*** és una càrrega apte per a qualsevol profunditat), sinó perquè **BLOC*** es redefineix, i cada vegada que canvii el valor d'un determinat atribut P en una inserció, això afectarà tant les insercions existents (sols caldrà sacsejar el dibuix fent ***REGEN***) com les que hagin de venir. Però si això constitueix un problema real (és a dir, si tenim atributs P i la intenció d'editar-ne), el sistema perquè deixi de ser-ho és ben elemental: n'hi ha prou a transferir també a **BLOC-2** els atributs P, i així haurem aplegat en un mateix sac tots els atributs editables, P i N, que pujaran al nivell superior, tornant a l'estat de preassignació, quan explosionem aquest bloc; dins de la inserció de **BLOC-1** només hi quedaran els constants, al costat de textos simples o múltiples i de la resta d'objectes gràfics. Tot depèn de si volem primar l'economia d'ocupació (que s'apreciarà en les insercions de **BLOC*** però no en les definicions, on els atributs P es manifesten d'una o altra manera, com a objectes **"ATTRIB"** que acompanyen la inserció de **BLOC-1** o com a objectes **"ATTDEF"**) o deixar que els atributs P siguin tan editables com els N.

De fet, la localització dels atributs P en **BLOC-1** i la dels atributs N en **BLOC-2**, més que respondre a un pla deliberat d'estalvi s'ha produït per una transposició mecànica i gairebé literal de l'expressió lògica

```
(if (and (= (cdr (assoc 0 LE)) "ATTDEF")
         (= (logand (cdr (assoc 70 LE)) 10) 0))
    ...)
```

des de la funció **ATRIBS-1** de la VERSIÓ 2 fins a la funció **SEGR-ATRIBS** de la VERSIÓ 8 i posteriors. I així com en el primer cas eren faves comptades (calia formar la llista **WWAA** de valors d'atributs N, per subministrar-los després a ***INSERT***), en el segon hauríem d'haver meditat les conseqüències d'emmagatzemar els atributs C i P en la llista **OO-1**, per crear **BLOC-1**, i els atributs N en la llista **OO-2**, per crear **BLOC-2** i explosionar-lo. Voluntat conscient o badada de principiant, allò que ara importa és que l'usuari pugui canviar de rumb sense tocar gaires tecles, i com que el codi ja està prou atapeït de línies de comentari amb opcions alternatives, serà millor confiar l'arbritratge a la variable booleana ***EDITATRIBS-P***, global en tota la sessió de dibuix, que guiarà el funcionament de **C:INSERTOK** i **C:DESCOMPOK** en una o altra direcció:

- En **C:INSERTOK** (funció **SEGR-ATRIBS**), l'expressió lògica esmentada incorporarà la nova variable i quedarà així

```
(if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
         (= (logand (cdr (assoc 70 LE))
                    (if *EDITATRIBS-P* 2 10)) 0))
    ...)
```

  Si, com havíem comentat a l'inici d'aquest capítol, a propòsit de la VERSIÓ 2, l'expressió **(= (logand (cdr (assoc 70 LE)) 10) 0)** només serà certa quan el valor associat al codi 70 d'un atribut (suma de les característiques ICVP compatibles) i el valor **10 = 2 + 8** no tinguin en comú cap dígit **1** en sistema binari, el canvi introdueix la possibilitat de mantenir aquesta condició (que ICVP no inclogui el sumand **2**-Constant ni el **8**-Predefinit), si ***EDITATRIBS-P*** és **nil**, o substituir-la per la menys estricta **(= (logand (cdr (assoc 70 LE)) 2) 0)** (que ICVP no inclogui el sumand **2**-Constant) si ***EDITATRIBS-P*** és **T**.
- En **C:DESCOMPOK**, només quan ***EDITATRIBS-P*** sigui **nil** necessitarem la intervenció de **PASSA-ATTRIB** per saltar-nos els atributs P (a diferència de la VERSIÓ 14+, no cal convertir aquesta rutina en funció, perquè **C:INSERTOK** ja no la utilitzarà). Presentant la variable en el seu context més immediat, ens ho trobarem així

```
(if (and (wcmatch BLOC* (strcat "*_" N))
         (setq SS (ssget "P")))
    (progn
      (setq E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))) N -1)
```

```
            (if (not *EDITATRIBS-P*)
                (while (wcmatch (cdr (assoc 0 (entget (entnext E*))))
                                "ATTRIB,SEQEND")
                       (setq E* (entnext E*))))
            (while (setq N (1+ N)
                         E (ssname SS N)
                         E* (entnext E*))
                   ...)))
```

L'assignació **(setq *EDITATRIBS-P* T)** la posarem al començament, perquè l'usuari la trobi fàcilment i la pugui deixar fora de servei, si s'estima més que els atributs P segueixin sent no editables.

Ja que estem amb ***-ATREDIT***, aprofitarem l'avinentesa registrant un estrany fenòmen que es dóna quan, després d'aplicar aquesta ordre a qualsevol bloc amb atributs, fent ús de l'opció ***P**osición*, executem **C:INSERTOK** en alguna de les circumstàncies següents: ser la primera vegada que la utilitzem amb un determinat **BLOC**; haver eliminat **BLOC-1**, **BLOC-2** i tots els genèrics **BLOC*** amb ***LIMPIA*** (després d'esborrar-ne les insercions, és clar); haver eliminat **BLOC-2** per forçar una redefinició de tots els **BLOC***s. En tots tres casos, l'execució s'interromp perquè ***-INSERT*** no ha pogut trobar **BLOC-1** o **BLOC-2**, en el procés de constitució d'un nou **BLOC*** (nou de trinca o redefinit), ben bé com si la funció **MAKEBLOC** (**entmake**, en definitiva) no respongués. Sense accés al codi font d'AutoCAD, ignorem el perquè d'aquests fets, però hem observat que el malefici s'esvaeix quan, entre ***-ATREDIT*** i la **C:INSERTOK** de pronòstic incert, s'executa una ordre ***INSERT*** (explícita o amagada dins d'una **C:INSERTOK** avaluada sense incidents, per fer ús d'algun **BLOC*** existent) o forcem una regeneració del dibuix. No necessitem saber més: com que la Medicina (inclosa la medicina preventiva) ha estat una ciència bàsicament empírica, ens limitarem a incloure **(command "REGENT")** en **SEGR-ATRIBS**, just abans dels accessos a **MAKEBLOC**. Tot i que els últims suggeriments es materialitzen en reformes o afegits locals a la VERSIÓ 13+, la seva dispersió fa aconsellable presentar el codi íntegre:

```
; VERSIÓ 15+

(setq *EDITATRIBS-P* T) ; Si voleu que els atributs predefinits siguin editables.

;;  (vl-load-com)
;;  (vlr-remove-all)
;;  (vlr-LISP-reactor () '((:vlr-LispWillStart . INSNAME-1)
;;                         (:vlr-LispEnded . INSNAME-2)))
;;  (vlr-COMMAND-reactor () '((:vlr-CommandWillStart . INSNAME-3)
;;                            (:vlr-CommandEnded . INSNAME-4)))
;;
;;  (defun INSNAME-1 (N-REACTIU L-ELISP)
;;    (if (= (substr (car L-ELISP) 1 17) "(C:INSERTOK)") (setq *INSNAME* 0)))
;;
;;  (defun INSNAME-2 (N-REACTIU L-ELISP)
;;    (setq *INSNAME* (if (= *INSNAME* 1) 0)))
;;
;;  (defun INSNAME-3 (N-REACTIU L-ORDRE)
;;    (if (= *INSNAME* 0) (setq *INSNAME* 1)))
;;
;;  (defun INSNAME-4 (N-REACTIU L-ORDRE)
;;    (if (= *INSNAME* 1)
;;        (setq *INSNAME* 0)
;;        (if (= *INSNAME* 0)
;;            (progn
;;              (setq *INSNAME* ())
;;              (setvar "INSNAME" *BLOC*)))))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
```

```
            (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
            " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun MAKEBLOC (BLOC/2 OO ATRIBS)
  (entmake (list '(0 . "BLOCK")
                 (cons 2 BLOC/2)
                 '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun LINIA-BASE ()
  (polar (cdr (assoc 11 LE))
         (- (cdr (assoc 50 LE)) PI/2)
         (* (cdr (assoc 40 LE))
            (if (= C-74 1)
                (/ 1.0 -3)
                (if (= C-74 2) 0.5 1)))))

(defun REDEF-BLOC*S (/ BLOK*)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (setvar "ATTREQ" 0)
  (while LB
         (if (wcmatch (setq BLOK* (cdr (assoc 2 LB))) N)
             (progn
               (setq E (cdr (assoc -2 LB))
                     LE (entget E)
                     J (cdr (assoc 41 LE))
                     K (/ (* (cdr (assoc 50 LE)) 180) PI))
               (command "INSERT" BLOC-1 '(0 0 0) J J K)
               (setq SS (ssadd (entlast)))
               (if OO-2
                   (progn
                     (command "INSERT" BLOC-2 '(0 0 0) J J K
                              "DESCOMP" (entlast))
                     (setq SA (ssget "P") K -1)
                     (while (setq E (ssname SA (setq K (1+ K))))
                            (ssadd E SS))))
               (command "BLOQUE" BLOK* "S" '(0 0 0) SS ""))
             (setq LB (tblnext "BLOCK")))
  (setvar "ATTREQ" (if ATREQ 1 0))
  (setvar "CLAYER" CAPA))

(defun SEGR-ATRIBS (/ BL1EX BL2EX AT AT-CP C-72 C-74 LB)
  (setq BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2))
  (if (and BL1EX BL2EX)
      (setq OO-2 (/= (cdr (assoc 0 (entget (cdr (assoc -2 BL2EX))))) "ENDBLK"))
      (progn
        (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
```

```
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (cdr (assoc 70 LE))
                                   (if *EDITATRIBS-P* 2 10)) 0))
                   (setq C-72 (cdr (assoc 72 LE))
                         C-74 (cdr (assoc 74 LE))
                         LE (if (= C-72 4)
                                (subst (cons 72 (setq C-72 1))
                                       (cons 72 4)
                                       (subst (cons 74 (setq C-74 2))
                                              (assoc 74 LE)
                                              LE))
                                LE)
                         LE (if (and (< C-72 3) (> C-74 0))
                                (subst (cons 74 0)
                                       (assoc 74 LE)
                                         (subst (cons 11 (LINIA-BASE))
                                                (assoc 11 LE)
                                                LE))
;;                                       (subst (cons 11 (LINIA-BASE))
;;                                              (assoc 11 LE)
;;                                              LE))
; Les 3 línies precedents poden usar-se com alternativa a les 5 línies subsegüents
                                       (if (> C-72 0)                          ;;
                                           (subst (cons 11 (LINIA-BASE))       ;;
                                                  (assoc 11 LE)                ;;
                                                  LE)                          ;;
                                           LE))                                 ;;
                                LE)
                         OO-2 (cons LE OO-2))
                   (setq OO-1 (cons LE OO-1)
                         AT-CP (if AT-CP T AT)))
               (setq E (entnext E)))
        (tblnext "BLOCK" T)
        (while (and (setq LB (tblnext "BLOCK"))
                    (not (setq BL*EX (wcmatch (cdr (assoc 2 LB)) N)))))
        (command "REGENT"
                 "CECOLOR" "PORCAPA"
                 "CELTYPE" "PORCAPA"
                 "CELWEIGHT" -1)
        (MAKEBLOC BLOC-1 OO-1 AT-CP)
        (MAKEBLOC BLOC-2 OO-2 OO-2)
        (if BL*EX (REDEF-BLOC*S))
        (setvar "CECOLOR" COL)
        (setvar "CELTYPE" TLIN)
        (setvar "CELWEIGHT" GLIN))))

(defun INSERT* (/ BLOC-1 BLOC-2 BLOC* BL*EX OO-1 OO-2 E LE SS SA X* X** Y**
                  OX* OX**)
  (setq W (if (< J K) (angle B X) (angle X B))
        B (mapcar '/ (mapcar '+ B X) '(2 2))
        W (angle O (polar B W (distance O B)))
        X* (inters X (polar X W 1) O B ())
        X** (inters O (polar O (+ W PI/2) 1) X X* ())
        Y** (inters Y (polar Y W 1) O X** ())
        OX* (distance O X*) OX** (distance O X**)
        BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
        BLOC* (strcat BLOC "_" (itoa (fix (/ OX** OX* Q0)))))
  (SEGR-ATRIBS)
  (if (not (setq BL*EX (tblsearch "BLOCK" BLOC*)))
      (progn
        (command "CLAYER" "0"
                 "CECOLOR" "PORBLOQUE"
                 "CELTYPE" "PORBLOQUE"
                 "CELWEIGHT" -2
                 "INSERT" BLOC-1 O (setq K (/ OX* OX**)) K X*)
        (setq SS (ssadd (entlast)))
        (if OO-2
            (progn
              (command "ATTREQ" 0
```

```
                        "INSERT" BLOC-2 O K K X*
                        "ATTREQ" (if ATREQ 1 0)
                        "DESCOMP" (entlast))
                (setq SA (ssget "P") K -1)
                (while (setq E (ssname SA (setq K (1+ K))))
                       (ssadd E SS))))
          (command "SCP" "Z" (setq K (trans O 1 0) O O) X**
                   "BLOQUE" BLOC* (trans K 0 1) SS ""
                   "SCP" "PR"
                   "CELWEIGHT" GLIN
                   "CELTYPE" TLIN
                   "CECOLOR" COL
                   "CLAYER" CAPA)))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) I) X**))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ BLOC N O X Y
                     OX OY XY A B W I J K)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (setq PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        O (getvar "INSNAME")
        O (if (wcmatch O (strcat "*_" N))      ;; Alternativa a l'ús de reactius
              (substr O 1 (- (strlen O) (strlen N) 1)) O)      ;; LISP i COMMAND.
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
;;        *BLOC* BLOC
        N (strcat BLOC "_" N)
        O (getpoint "\nPrecise punto de inserción: "))
  (foreach P '("X" "Y")
        (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                    " desde el punto de inserción: "))
              I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                   BLOC " <1>: "))
              W (angle O A)
              B (/ (distance O A) (if I I 1)))
        (set (read P) (polar O W B))
        (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY)
        B (polar O (- W PI/2) OY)
        XY (distance X Y)
        I (if (< (distance A X) (distance B X)) -1 1))
  (setvar "CMDECHO" 0)
  (setvar "OSMODE" 0)
  (setvar "ATTDIA" 0)
  (if (not (or (equal XY (+ OX OY) Q0) (equal XY (abs (- OX OY)) Q0)))
      (if (equal (setq J (expt XY 2)) (setq K (+ (expt OX 2) (expt OY 2))) Q0)
          (command "INSERT" BLOC O OX (* OY I) X)
          (INSERT*)))
  (prompt (if (> (getvar "CMDACTIVE") 0) "\nIndique valores de atributo\n" "\n"))
  (setvar "ATTDIA" ATDIA)
  (setvar "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))
```

```
(defun C:DESCOMPOK (/ Q0 ECO BLOC* E E* LE SS A N)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INSERTOK, sin deformación de ")
  (prompt "sus atributos")
  (prompt "\n(si se definieron con caracteres oblicuos, usando DESCOMP se ")
  (prompt "reduce la altura):")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              ECO (getvar "CMDECHO"))
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (setvar "CMDECHO" ECO)
        (if (and (wcmatch BLOC* (strcat "*_" N)) (setq SS (ssget "P")))
            (progn
              (setq E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))) N -1)
              (if (not *EDITATRIBS-P*)
                  (while (wcmatch (cdr (assoc 0 (entget (entnext E*))))
                                  "ATTRIB,SEQEND")
                         (setq E* (entnext E*))))
              (while (setq N (1+ N)
                           E (ssname SS N)
                           E* (entnext E*))
                     (setq A (cos (cdr (assoc 51 (entget E*)))))
                     (if (not (equal A 1 Q0))
                         (progn
                           (setq LE (entget E)
                                 LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A))
                                           (assoc 40 LE) LE)
                                 LE (subst (cons 41 (* (cdr (assoc 41 LE)) A))
                                           (assoc 41 LE) LE))
                           (entmod LE)))))))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

Acabarem el capítol confessant un pecat de manca de rigor (venial, això sí) comès en les dues últimes versions i que fins ara havíem cobert amb un silenci culpable: a la VERSIÓ 14+, quan la funció **CORR-POSICIO** calculava el desplaçament lateral **OI** dels atributs a partir d'una distància **H** entre el punt d'ancoratge i la línia base que, en el cas d'utilització del mode de justificació ***M**edio*, assimilàvem a la dels modes ***M**edio-**I**zquierda*, ***M**edio-**C**entro* i ***M**edio-**D**erecha*, tot considerant-la la meitat de l'alçada de text; a la VERSIÓ 15+, quan **SEGR-ATRIBS** començava transformant els atributs amb mode de justificació ***M**edio* a justificació ***M**edio-**C**entro*, mantenint el punt d'ancoratge, i tot seguit transformava tots els d'aquest últim mode a ***C**entro*, substituint el punt d'ancoratge pel que volia ser (però no sempre era) la seva projecció sobre la línia base (**LINIA-BASE** es limitava a considerar un desplaçament igual a la meitat de l'alçada de text, en direcció perpendicular a aquesta línia). La fal·làcia consisteix a suposar que la distància entre el punt d'ancoratge i la línia base, quan tenim justificació ***M***, és constant: sí que ho és, per definició, quan tenim ***MC***, però ***M*** no considera l'alçada de text (distància entre la línia base i la línia superior, hi hagi o no en el text caràcters que freguin aquesta última) sinó l'alçada efectiva del text assignat a l'atribut en cada cas (és a dir, la distància entre l'extrem inferior del caràcter que arribi més avall i l'extrem superior d'aquell que arribi més amunt).

En aquest sentit, si ens limitem a identificadors i valors d'atribut constituïts exclusivament per caràcters alfanumèrics, prescindint de majúscules accentuades o amb dièresi, aquests es podrien classificar en quatre grups:

A) Nombres, lletres majúscules tret de "Ç", i determinades minúscules: "b", "d", "f", "h", "i", "k", "l", "ñ", "t" i vocals accentuades o amb dièresi.
B) Les lletres minúscules "a", "c", "e", "m", "n", "o", "r", "s", "u", "v", "w", "x" i "z".
C) Les lletres minúscules "ç", "g", "p", "q" i "y".
D) Les lletres "j" i "Ç".

Si l'estil de text no utilitza un tipus de lletra "True Type" sinó de forma (arxiu .SHX), l'alçada dels caràcters del grup A coincideix amb allò que anomenem alçada de text (tota per sobre de la línia base), la del grup B és 2/3 d'aquesta alçada (tota per sobre de la línia base), la de C també hi coincideix (tot i que 2/3 de l'alçada se situa per sobre de la línia base i 1/3 per sota) i la de D és de 1 1/3 (una alçada sencera per sobre de la línia base i 1/3 per sota). I, pel que fa a la cadena de caràcters:

- Un text compost només per caràcters del grup A i del grup B es comportarà com si estigués compost exclusivament per caràcters del grup A, i ***M*** i ***MC*** coincidiran (a 1/2 de l'alçada de text, per damunt de la línia base).
- En un text compost exclusivament per caràcters del grup B, el punt ***M*** se situarà 1/6 de l'alçada de text per sota de ***MC*** (a 1/3 de l'alçada de text, per sobre de la línia base).
- En un text on hi ha el caràcter "Ç" o que està compost per caràcters del grup A i del grup C (tant se val si també n'hi ha del grup B), el punt ***M*** se situarà 1/6 de l'alçada de text per sota de ***MC*** (a 1/3 de l'alçada de text, per sobre de la línia base).
- Un text compost només per caràcters del grup B i del grup C es comportarà com si estigués compost exclusivament per caràcters del grup C, i ***M*** se situarà 1/3 de l'alçada de text per sota de ***MC*** (a 1/6 de l'alçada de text, per sobre de la línia base).

En la VERSIÓ 14+, assimilar el mode de justificació ***M*** al mode ***MC***, pel que fa a la determinació del desplaçament lateral d'un atribut i just quan crèiem que anàvem a contrarestar-lo, pot provocar-ne un d'amplitud 1/6 o 1/3 de la prevista i de signe contrari. Això es podria evitar complicant una mica la funció **CORR-POSICIO**, on en comptes figurar el valor fix **0.5** a la quarta línia podia haver-hi la crida a una nova funció **ALT-ATTRIB** que analitzaria cada cas i determinaria la fracció d'alçada de text en funció del valor concret assignat a l'atribut (**(cdr (assoc 1 LE)**):

```
(defun ALT-ATTRIB (/ V L M N A B C D)
  (setq V (cdr (assoc 1 LE)) N 0)
  (repeat (strlen V)
    (if (not D)
        (progn
          (setq N (1+ N) L (substr V N 1))
          (if (setq M (wcmatch L "[jÇýÿ]"))
              (setq D M)
              (if (setq M (wcmatch L "[çgpqyµ]"))
                  (setq C M)
                  (if (setq M (wcmatch L "[:;acemnorsuvwxzæø]"))
                      (setq B M)
                      (if (setq M (wcmatch L "~[ `-`,`.=·÷]"))
                          (setq A M))))))))
  (if (or D (and A C) (and (not A) B (not C)))
      (/ 1.0 3)     ; qualsevol combinació amb D, o bé ABC, AC o B
      (if C
          (/ 1.0 6) ; BC o C
          0.5)))    ; A, AB o no res

(defun CORR-POSICIO (/ H A SINA COSA TANA OI)
  (setq H (* (cdr (assoc 40 LE*))
             (if (= C-72 4)
                 (ALT-ATTRIB)
                 (if (= C-74 1)
                     (/ 1.0 -3)
                     (if (= C-74 2) 0.5 1))))
        A (cdr (assoc 50 LE*)) SINA (sin A) COSA (cos A)
        TANA (if (not (equal COSA 0 Q0)) (abs (/ SINA COSA)))
        OI (if TANA (* H (sqrt (+ (expt (* EX SINA) 2) (expt (* EY COSA) 2)))
                       (cos (+ (atan (* EY/EX TANA)) (atan (/ EY/EX TANA)))))
                    0))
  (polar (cdr (assoc 11 LE)) (cdr (assoc 50 LE)) OI))
```

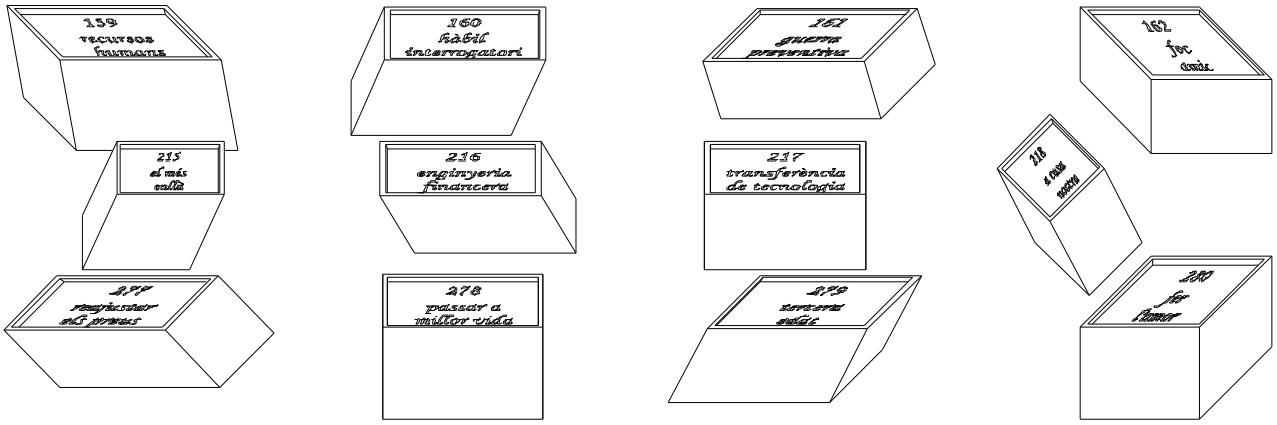
Malgrat haver considerat tots els caràcters utilitzables (32 al 126 del codi ASCII bàsic, i del 160 al 255 del codi ASCII estès per a Windows), més enllà dels que havíem explicitat, cal advertir que aquesta versió de **ALT-ATTRIB** només funcionaria

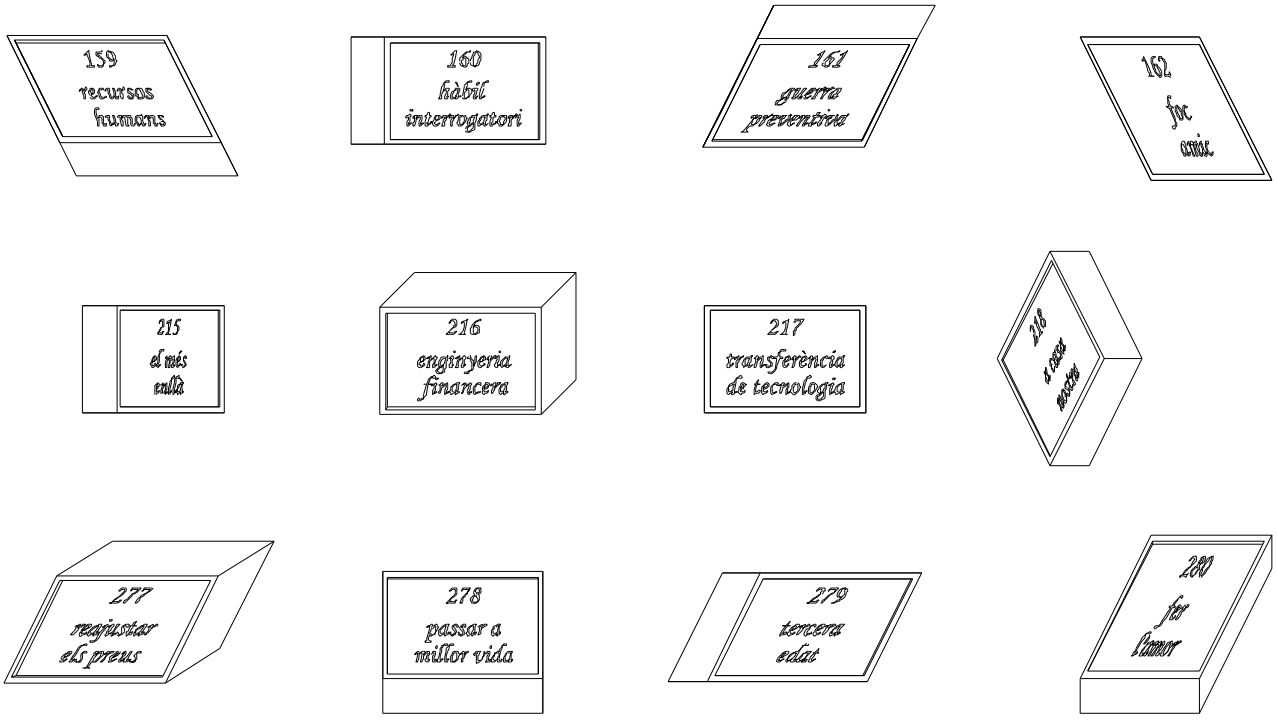
raonablement bé amb el tipus de lletra TXT.SHX (caràcters com "*", "+", ",", ":", ";", "_" i altres, en què l'excés o defecte respecte a les línies base o superior no arriba a 1/3 de la distància entre ambdues (sols és 1/6), encara podrien donar lloc a petits desplaçaments, però ho hem deixat així per no complicar la funció). Si a més volguéssim adaptar-la a d'altres fonts .SHX (SIMPLEX, COMPLEX, o bé les famílies GHOTIC*, GREEK*, ISO*, ITALIC*, ROMAN* o SCRIPT*) en què les majúscules accentuades o amb dièresi no sempre s'ajusten a la línia superior, o en què "()", "[]", "{}" i d'altres caràcters no s'ajusten ni a la línia base ni a la superior, gairebé no podríem generalitzar i hauríem de seguir un mètode casuístic poc àgil.

De tota manera, com que els avantatges de la VERSIÓ 15+ ja ens havien decidit a considerar superada la precedent, haurem d'afrontar els efectes indesitjats que introduirà en aquesta el fet d'assimilar el mode de justificació ***M*** a ***MC***, amb el mateix punt d'ancoratge, i tot seguit substituir ***MC*** per ***C***, amb el punt d'ancoratge sobre la línia base. Fèiem la suplantació com a recurs econòmic per garantir que no hi hauria desplaçaments laterals, però allò que no podrem evitar en determinats casos serà un desplaçament cap avall de 1/6 o 1/3 l'alçada de text, que si voleu mai no serà tan aparatós, de cara a la percepció d'estar un text centrat amb els components gràfics que l'envolten, però que no deixa de ser una deficiència en relació als objectius fixats.

Dir que sempre podrem ajustar *a posteriori* la posició dels atributs desplaçats cap avall (amb ***-ATREDIT*** però manualment, perquè integrar la correcció en el programa seria tant com tornar a la VERSIÓ 14+) només és un atenuant; mai serà un eximent. I quan d'aquí endavant diguem que els modes de justificació ***M****edio* i ***M****edio-****C****entro* són "gairebé equivalents", ja sabreu que l'adverbi no és tan innocent...

SMES GALERIA D'EUFEMISMES GALE
ECTES POLÍTICAMENT CORRECTES POLÍT
ÍCULS I PREGONAMENT RIDÍCULS I PRE





ENCAIX 3D DE BLOCS AMB ATRIBUTS

En els dos capítols precedents ens hem ocupat de com manipular un bloc per tal que el seu quadrat ortonormal de referència encaixés en un paral·lelogram qualsevol. Enteníem que era un bloc 2D, és a dir, compost exclusivament per objectes plans i situats en el seu pla ***XY*** (si no era així les dimensions ***z*** no ens importaven gens), i també acceptàvem implícitament que el paral·lelogram d'encaix era horitzontal (paral·lel al pla ***XY*** del sistema de coordenades vigent durant la inserció). Ara donarem un pas endavant, per ocupar-nos de blocs amb una dimensió ***z*** significativa i de com manipular-los per tal que el seu cub ortonormal de referència encaixi en un paral·lelepípede lliurement orientat. Pel que fa a la composició del bloc, no perdrem el temps bastint una primera versió només vàlida per a blocs desproveïts d'atributs: en 2D ho havíem fet així per no sentir-nos aclaparats i poder centrar-nos en la problemàtica geomètrica, però ara que tenim resoltes (de moment) les dificultats degudes a la presència d'atributs editables, ya no farem marxa enrera.

De primer, però, abordarem el vessant geomètric que, com a NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, contemplarà l'anàlisi geomètrica pròpiament dita i la discussió, d'ordre més pràctic, sobre els factors d'escala que convé utilitzar en cada inserció. Com és lògic, anirem per feina i evitarem ser redundants en relació a l'esmentat capítol, repetint consideracions que ja haguéssim fet allà o fent-ne d'altres que en siguin directament deduïbles. Sols ens permetem recordar al lector la disparitat de referents entre els subíndexs **x**, **y** (i ara **z**) aplicats als factors d'escala **E** i les majúscules **X**, **Y** (i **Z**), que en cada transformació representen els punts **P** definitoris del cub transformat en paral·lelepípede, del cub original (**P′**) i de la seva proyecció sobre l'eix d'afinitat considerat (**P″**): en cada inserció, la direcció **x** dels primers es correspon amb aquest eix (**O–P″**) i la direcció **y** amb la d'afinitat (**P″–P′–P**).

Malgrat que la filosofia és la mateixa, tot se'ns complica per la reiteració del procés executiu (definició i inserció de blocs), de dues etapes
1- Inserció de **BLOC** i conversió en **BLOC***
2- Inserció de **BLOC***,
que ara queda configurat esquemàticament en tres:
1- Inserció de **BLOC** i conversió en **BLOC***
2- Inserció de **BLOC*** i conversió en **BLOC****
3- Inserció de **BLOC****.

Si prenem com a referències el cub **1 × 1 × 1** associat a **BLOC** i el seu transformat al final del procés, que en el cas més general serà un paral·lelepípede oblic que considerarem estructurat en dos plans (el de la base **X–O–Y** i un de perpendicular per l'aresta **O–Z**), en una primera aproximació podem descriure aquestes etapes de la manera següent:

1- Inserció de **BLOC** en **O**, amb $\mathbf{E_x = E_y = E_z}$ (escalat uniforme) i gir (dos gir al voltant d'eixos perpendiculars, un d'ells integrat a ***INSERT*** i l'altre a càrrec de l'ordre ***GIRA***), convertint el resultat en **BLOC***. El cub de referència de **BLOC** seguirà sent un cub a **BLOC*** perquè els continguts són semblants geomètricament.

2- Inserció de **BLOC*** en **O**, amb $\mathbf{E_x \neq E_y \neq E_z}$ i gir, feta de manera que el cub de referència es transformi en un prisma oblic, de bases encara quadrades però situades ja en els mateixos plans que les bases del paral·lelepípede oblic final (***INSERT*** des d'un ***SCP*** amb el pla ***XY*** perpendicular al de la base **X–O–Y**). Conversió del resultat en **BLOC****.

3- Inserció de **BLOC**** en **O**, amb $\mathbf{E_y \neq E_x = \pm E_z}$ i gir, feta de manera que el prisma oblic de referència en **BLOC**** es transformi en el paral·lelepípede oblic final: la base quadrada en el paral·lelogram **X–O–Y**, i **Z** prenent la posició definitiva (***INSERT*** des d'un ***SCP*** amb el pla ***XY*** coincident amb el de la base **X–O–Y**).

La millor manera d'aprofundir en el procés serà recórrer-lo ara en sentit invers, des del paral.lelepípede oblic final fins al cub ortogonal **1 × 1 × 1** associat al bloc original, afinant-ne la descripció, precisant posicions i angles, i enumerant els diversos sistemes de referència. Aquest itinerari **3 → 2 → 1** no és unicament un recurs explicatiu, sino que respon al procés de recollida i elaboració de dades

(que té com a objectiu calcular els paràmetres que haurem d'utilitzar després), previ al procés executiu **1 → 2 → 3** que acabem de presentar (definició, si encara no existien, i inserció de **BLOC*** i **BLOC****):

3- La transformació d'un prisma oblic de base quadrada **O–X′ × O–Y′ × O–Z′** en un paral·lelepípede oblic **O–X × O–Y × O–Z** de igual alçada (en ser coplanàries les bases inferiors, que comparteixen el vèrtex **O**, també ho seran les superiors) és la mateixa que havíem analitzat en el capítol NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, pel que fa a les bases inferiors: així doncs, realitzarem la construcció de Ritz per trobar l'eix d'afinitat **X″–O–Y″** i les dimensions i la posició del quadrat **X′–O–Y′**, a partir del paral·lelogram **X–O–Y**; i, de moment, suposarem que el bloc **BLOC**** té com a prisma oblic de referència el descrit (base quadrada i l'alçada del paral·lelepípede final, sense escalat addicional) i que l'inserim amb els factors $\mathbf{E_x = E_z = 1}$ i $\mathbf{E_y = \frac{Y''–Y}{Y''–Y'}}$. Remuntant-nos ara al pla de les bases superiors, veurem com la recta **Z–Z′** (**O–Z** no és perpendicular a la base) és paral·lela a **X–X′** i a **Y–Y′**. Si anomenem **Z″** la intersecció de **Z–Z′** amb el pla normal a les bases i que passa per l'eix d'afinitat **X″–O–Y″**, també s'acomplirà que $\mathbf{\frac{Z''–Z}{Z''–Z'} = \frac{X''–X}{X''–X'} = \frac{Y''–Y}{Y''–Y'}}$ y podrem localitzar **Z′** sobre **Z–Z″** calculant la distància $\mathbf{Z''–Z' = \frac{X''–X'}{X''–X} \times Z''–Z}$. Ara ja disposem dels paràmetres definitoris de l'etapa 3, dades de partença necessàries per determinar els de l'etapa 2, però abans de retrocedir fins aquesta farem un parell d'advertiments:

- El primer és per destacar que, tant l'avaluació d'aquests paràmetres com la prèvia qualificació del paral·lelepípede d'encaix (base quadrada o romboïdal, prisma recte o oblic) es fa sota un ***SCP*** d'origen **O** (tots els que utilitzem a partir d'ara tindran el mateix origen), eix ***X*** coincident amb l'aresta **O–X** d'aquest paral·lelepípede i, com a pla coordenat ***XY***, el de la base **X–O–Y**. Anomenarem "A" aquest sistema de referència.
- El segon es refereix a la sintaxi que utilitzarem a partir d'ara, que haurem de considerar la construcció de Ritz en un pla perpendicular a l'esmentat i que passa per **O–Z′**: la posició **Z′** la notarem **(Z′)** per poder seguir aplicant els mateixos criteris semàntics en aquest altre pla, i parlarem dels punts $\mathbf{(Z')'}$ i $\mathbf{(Z')''}$ en comptes de **Z″** i **Z‴** (polisèmics tots dos i excessiu el segon); així, **X**, **X′**, **X″**, **Y**, **Y′** i **Y″** seran a partir d'ara **(X)**, **(X′)**, **(X″)**, **(Y)**, **(Y′)** i **(Y″)** respectivament, i, considerant la intersecció entre els dos plans esmentats, anomenarem **(XY′)** el punt d'aquesta recta situat a una distància **O–(X′)** de **O**, a la banda on talli la diagonal **(X′)–(Y′)** del quadrat de referència (si **O–(XY′)** i **(X′)–(Y′)** són paral·leles, a la mateixa banda on se situa **(Z′)**).

2- La transformació d'un cub en el prisma oblic de referència del bloc **BLOC**** que només conserva quadrades les bases, també és la mateixa que havíem analitzat en NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, pel que fa al pla determinat per **O–(Z′)** i **O–(XY′)** (aquest pla, que passa per l'aresta **O–(Z′)** del prisma i és perpendicular a la base, coincideix amb el que passa per l'aresta $\mathbf{O–(Z')'}$ del cub i ho és a la seva), per bé que fora d'aquest pla introdueix una distorsió que caldrà contrarestar amb un escalat reductor en la direcció normal. Però també a propòsit de la construcció de Ritz, que en l'etapa 3 aplicàvem sense reserves, aquí convindrà fer un aclariment: el procediment es basava en la premissa que la figura a transformar era un quadrat; però en aquest pla transversal només les arestes homòlogues $\mathbf{O–(Z')'}$ i **O–(Z′)** són elements rellevants ja que, de les altres dues $\mathbf{O–(XY')'}$ i **O–(XY′)**, l'única informació significativa és l'orientació. En efecte, no tindria cap sentit que ens entestéssim a calcular la secció del prisma oblic pel pla perpendicular a la base des de **O–(Z′)**: fora del cas que la recta **O–(XY′)** passés per **(X′)** o per **(Y′)**, quan tallés la diagonal **(X′)–(Y′)** entre aquests dos punts tidríem secció real, un paral·lelogram de base **O–(W′) > O–(X′)**, i la secció del cub de referència de **BLOC*** pel mateix pla seria un rectangle apaïsat, de base $\mathbf{O–(W')'} > \mathbf{O–(Z')'}$, i tanmateix sabem que el mètode gràfic només

és aplicable si $O\text{–}(W')' = O\text{–}(Z')'$; això sense parlar dels casos sense secció real, per ser el pla paral·lel a la diagonal $(X')\text{–}(Y')$ o per situar-se la intersecció més enllà d'aquests punts. En qualsevol cas, sempre podem suposar que, en lloc d'un cub que es transforma en prisma oblic, manipulem un cilindre de revolució que es transforma en un d'oblic, d'eix $O\text{–}(Z')$ i base un cercle de radi $O\text{–}(XY')$. Per tot això, tirarem pel dret i considerarem el paral·lelogram $(XY')\text{–}O\text{–}(Z')$ (per construcció és $O\text{–}(XY') = O\text{–}(X')$) i un quadrat $(XY')'\text{–}O\text{–}(Z')'$ que legitimi l'aplicació del mètode: seguirem la construcció de Ritz per trobar, a partir de l'esmentat paral·lelogram, l'eix d'afinitat $(XY')''\text{–}O\text{–}(Z')''$ i les dimensions $O\text{–}(XY')' = O\text{–}(Z')'$ i posició del quadrat. Anomenarem "B" el sistema de referència en el pla ***XY*** del qual té lloc la construcció, que haurem obtingut a partir del "A" mitjançant dos ordres ***SCP***: la primera amb un gir respecte a l'eix ***Z***, per situar l'eix ***X*** damunt de $O\text{–}(XY')$; la segona amb un gir de $\pm 90^{\circ}$ respecte al nou ***X***. De moment, considerem que aquest cub de $O\text{–}(XY')' \times O\text{–}(XY')' \times O\text{–}(XY')'$ és el cub de referència de **BLOC***, i que l'inserim amb els factors $E_x = 1$, $E_y = \dfrac{(Z')''\text{–}(Z')}{(Z')''\text{–}(Z')'}$ i $E_z = \dfrac{O\text{–}(X')}{O\text{–}(XY')'}$. Abans de retrocedir fins a l'etapa 1, però, haurem de justificar perquè ara no és $E_x = E_z$: si un quadrat situat en el pla determinat per $O\text{–}(XY')'$ i per l'eix ***Z*** del sistema "B" (la base del cub, de $O\text{–}(XY')' \times O\text{–}(XY')'$) s'ha de transformar en un altre quadrat situat en el pla ***ZX*** del mateix sistema (la base del prisma oblic, de $O\text{–}(XY') \times O\text{–}(XY')$), el transformat de qualsevol punt $(P')'$ del primer ha de ser un punt (P') del segon en què la raó $\dfrac{O\text{–}(P')}{O\text{–}(P')'}$ sigui constant; pel que fa als punts $(XY')'$ i (XY') situats en el pla on realitzem la construcció, coneixem el valor d'aquesta constant, que és $\dfrac{O\text{–}(XY')}{O\text{–}(XY')'} = \dfrac{O\text{–}(X')}{O\text{–}(XY')'}$, o sigui que en la direcció normal (eix ***Z***) caldrà aplicar $E_z = \dfrac{O\text{–}(X')}{O\text{–}(XY')'}$.

1- Considerem la transformació del cub coordenat $1 \times 1 \times 1$, referència del bloc original **BLOC**, en el cub de referència del bloc genèric de primer grau **BLOC*** que s'insereix en l'etapa 2, i deixem per a després la discussió dels factors d'escala (de moment, n'hi ha prou a saber que $E_x = E_y = E_z$), per centrar-nos ara en la problemàtica de la seva orientació. Però abans que res cal insistir en el fet que la determinació dels paràmetres de les transformacions que se succeeixen a partir de l'etapa inicial, i que ja hem descrit en aquesta visió retrospectiva, ha de ser prèvia i s'ha de fer precisament en l'ordre invers: començant per l'etapa 3 ($(X)\text{–}O\text{–}(Y) \rightarrow (X'')\text{–}O\text{–}(Y'')$, $(X) \rightarrow (X')$, $(Y) \rightarrow (Y')$, $(Z) \rightarrow (Z')$) i passant a la 2 ($(XY')\text{–}O\text{–}(Z') \rightarrow (XY')''\text{–}O\text{–}(Z')''$, $(XY') \rightarrow (XY')'$, $(Z') \rightarrow (Z')'$). Si no, no podríem orientar correctament **BLOC**, aplicant-li els dos girs necessaris per deixar-lo en la posició de partença de l'etapa 2, convertit ja en **BLOC***. Els dos girs, d'eixos perpendiculars que es tallen en **O**, són:

- Des del sistema que hem anomenat "A", el d'angle $O\text{–}(X')$ (aquest gir durà la semirecta $O\text{–}(X)$ a situar-se sobre $O\text{–}(X')$).
- Des del sistema que hem anomenat "B", el d'angle $O\text{–}(XY')'$ (aquest gir durà la semirecta $O\text{–}(XY')$ a situar-se sobre $O\text{–}(XY')'$).

Aquesta manipulació la podríem dur a terme en quatre operacions:

- Ordre ***SCP*** per situar-nos en el sistema "A", i ***INSERT*** per inserir **BLOC** amb els factors $E_x = E_y = E_z$ que de seguida discutirem, sense fer-lo girar.
- Ordre ***GIRA*** per fer girar la inserció de **BLOC** el primer angle.
- Dues ordres ***SCP*** per situar-nos en el sistema "B".
- Ordre ***GIRA*** per fer girar la inserció de **BLOC** el segon angle (amb ***GIRA3D*** ens podríem haver estalviat les ordres ***SCP*** però, com veurem de seguida, ja ens convé accedir al sistema "B", de camí cap a un nou sistema "B2").

Però complicar les coses, realitzant el primer gir al marge de la inserció, no reporta cap avantatge. Així que ho resoldrem en tres operacions:
- Ordre ***SCP*** per situar-nos en el sistema "A", i ***INSERT*** per inserir **BLOC** amb els factors $\mathbf{E_x = E_y = E_z}$ i fer-lo girar el primer angle.
- Dues ordres ***SCP*** per situar-nos en el sistema "B".
- Ordre ***GIRA*** per fer girar la inserció de **BLOC** el segon angle.

Ja només restarà executar de nou ***SCP*** perquè el segon sistema giri al voltant de l'eix ***Z***: es manté el pla coordenat ***XY***, però ara l'eix ***X*** coincidirà amb $\mathbf{O\text{–}(XY')''}$. En aquest nou sistema de referència, que anomenarem "B2" i en què el quadrat $\mathbf{O\text{–}(XY')' \times O\text{–}(XY')'}$ quedarà inclinat un angle de cosinus $\frac{\mathbf{O\text{–}(XY')''}}{\mathbf{O\text{–}(XY')'}}$, definirem **BLOC***.

Pot causar certa perplexitat que haguem definit **BLOC*** en el sistema "B2", que comparteix l'eix ***Z*** amb el "B" on hem efectuat el segon gir, i que tanmateix la inserció de **BLOC** (amb uns factors d'escala destinats a garantir que l'aresta del cub de referència de **BLOC*** faci $\mathbf{O\text{–}(XY')'}$, com veurem tot seguit, per poder aïllar en el pla ***XY*** comú els elements geomètrics que en l'etapa 2 ens hauran de permetre passar a **BLOC****) es faci des d'un sistema "A" d'eix ***Z*** perpendicular a l'anterior, per poder-hi realitzar el primer gir. Tot plegat, potser seria més entenedora una seqüència d'operacions on els papers representats pel genèric **BLOC*** es dividissin i repartissin entre dos, **BLOC+** i **BLOC++**, d'aquesta manera:
- Ordre ***SCP*** per situar-se en el sistema "A", i ***INSERT*** per inserir **BLOC** amb els factors $\mathbf{E_x = E_y = E_z = 1}$ i fer-lo girar el primer angle.
- Dues ordres ***SCP*** per situar-se en el sistema "B", i ***BLOQUE*** per convertir la inserció de **BLOC** en el genèric **BLOC+**.
- ***INSERT*** per inserir **BLOC+** amb uns altres factors $\mathbf{E_x = E_y = E_z}$ i fer-lo girar el segon angle.

Només restaria passar al sistema "B2" per definir-hi un **BLOC++** coincident amb **BLOC*** geomètricament (no tant pel contingut, ja que **BLOC++** seria una inserció d'inserció de **BLOC**). Però hem d'evitar recels infundats, fora de preferències atàviques per certes simetries formals: de primer renunciàvem a la possibilitat d'integrar en l'ordre ***INSERT*** el gir en el sistema "A", per tal que en el procés apareguessin dues ordres ***GIRA*** (l'una actuant en el sistema "A" i l'altra en el "B"); i ara hem estat a punt de passar-nos a l'altre extrem (al preu d'un bloc intermedi **BLOC++**) per tal que ambdós girs quedessin integrats en ordres ***INSERT***. Desempalleguem-nos de prejudicis estètics, i acceptem que la caracterització de **BLOC*** és perfectament legítima i que, si amb alguna qüestió objectiva tenen a veure les reticències manifestades, és amb la disjuntiva entre una estructura de genèrics compactada a 2 nivells (**BLOC*** i **BLOC****) o una sense compactar de 4 nivells (**BLOC+**, **BLOC++**, **BLOC+++** i **BLOC++++**), tema que reprendrem de seguida.

Abans de seguir, seria bo que fessim una pausa per reflexionar sobre un tema que des de bon començament havíem presentat com a indiscutible: el de l'estructura de genèrics formada per **BLOC*** (inserció de **BLOC** en què el cub de referència segueix sent un cub, generalment amb l'eix ***Z*** no vertical) i **BLOC**** (inserció de **BLOC*** en què el cub de referència s'ha convertit en un prisma, generalment oblic, de base quadrada en posició horitzontal), que tal vegada serà l'òptima però que per ara segueix sent una decisió sense justificar. En primer lloc, hauríem de relativitzar una afirmació feta dos capítols enrere: "Per optimitzar el sistema caldrà que el conjunt dels blocs genèrics de primer grau (**BLOC***) sigui el més reduït possible". Si en 2D (on només hi havia genèrics de primer grau) això era tautològic, més que incontestable, en 3D, amb dos nivells de genèrics (**BLOC*** i **BLOC****) s'hauria de matisar i substituir per la frase que trobem poc després: "A la Figura 8.1 es fa palès com la tendència a ajornar la informació específica, desplaçant-la cap a nivells més pròxims a les insercions finals (en paral·lel a un procés d'ampliació del perfil dels blocs a cada nivell), com en les solucions de la dreta, redunda en un sistema menys redundant" (el subratllat, que és d'ara, reflecteix el que volem expressar). Efectivament, la meitat inferior d'aquella figura (corresponent a 3D) podia induir a una generalització prematura (ens ho havíem fet anar massa bé), i n'hi ha prou a veure el contraexemple que mostra la Figura 10.1 per adonar-nos que no ha de ser necessàriament així: allà teníem a l'esquerra 4 genèrics de primer grau i 8 de segon (en total, 12 genèrics derivats de **BLOC** per servir 16 insercions dobles concretes), i a la dreta 2 genèrics de primer grau i 4 de segon (en total, 6 genérics per servir les mateixes insercions), mentre que aquí tenim a l'esquerra 3 genèrics de primer grau i 6 de segon (en total, 9 genèrics derivats de **BLOC** per servir 24 insercions triples concretes), i a la dreta 2 genèrics de primer grau i 8 de segon (en total, 10 genérics per servir les mateixes insercions).

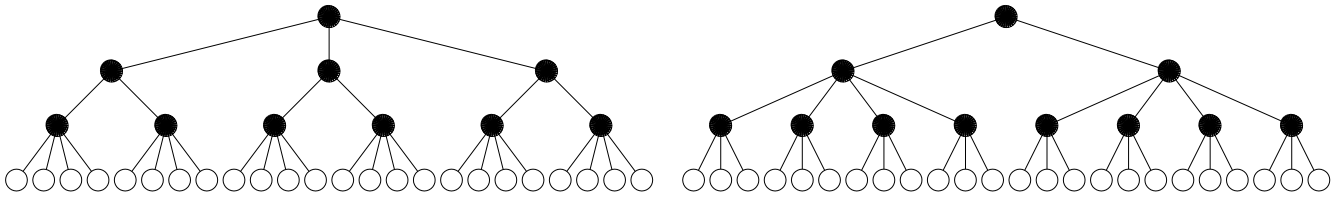


Figura 10.1

I és que no n'hi ha prou a desplaçar l'especificitat des del primer nivell fins al segon si els seus genèrics en tenen poc, de caràcter genèric, i cal que n'hi hagi més per cumplimentar totes les insercions, sinó que s'ha de tendir a que tots ells (entre el bloc primitiu i les insercions finals) siguin quan més genèrics millor. No s'hi val a dir que, portant al límit aquesta recomanació, el millor fóra passar directament de l'arrel **BLOC** (el més genèric dels blocs possibles) a les insercions triples corresponents a tots els paral·lelepípedes d'encaix previstos, perquè és una fal·làcia per partida doble: en primer lloc, perquè no només compta el total d'elements a movilitzar sinó la quantitat de paràmetres a què cada genèric estigui lligat, i cadascuna d'aquestes fulles estaria farcida d'informació redundant (sota diferents noms, es repetirien les definicions de **BLOC*** i **BLOC****); i en segon lloc, perquè la recomanació (desplaçar l'específitat cap a les fulles o, a la recíproca, desplaçar la "genericitat" cap a l'arrel) se situa en el context d'arbres amb un determinat nombre de nivells.

En efecte, no hem de barrejar criteris: quan analitzem diversos models en arbre, susceptibles de representar solucions alternatives per assolir l'encaix d'un bloc en una sèrie de paral·lelepipedes, per avaluar-ne l'economia de recursos, una cosa és comparar estructures jerarquitzades segons un mateix nombre de nivells (aquest és el context en què fins ara ens hem mogut) i una altra de molt diferent comparar estructures de diferents nivells. En aquest segon terreny tot són faves comptades: a partir dels nivells determinats pel mínim nombre d'insercions amb què es pugui resoldre el procés, tot excés serà redundant (fragmentar en diverses seqüències ***BLOQUE*** + ***INSERT*** allò que s'hagués pogut resoldre en una de sola és innecessari) i tot defecte serà artificiós (refondre en un únic bloc el producte d'una seqüència ineludible d'insercions és augmentar les probabilitats que la part inicial de la seqüència es repeteixi en una altra branca i, en definitiva, també és redundant). Un exemple de la primera desviació formalista (forçar la diferenciació allà on no cal) seria la possibilitat abans esmentada de desdoblar **BLOC*** en **BLOC+** i **BLOC++**, i **BLOC**** en **BLOC+++** i **BLOC++++**, i un exemple de la desviació oposada (donar aparença uniforme a allò que és intrinsecament divers) seria la intervenció explícita d'un únic nivell de genèrics entre **BLOC** i el conjunt de les insercions 3D adaptades a paral·lelepípedes diversos (anomenant **BLOC#**s als membres d'aquesta generació única d'intermediaris, coincident amb la dels nostres **BLOC****s, en seva la composició hi intervindria sempre la inserció del bloc previ **BLOC***, la definició del qual, no sent pública, s'hi aniria repetint cada vegada sota noms diferents).

Tot i així, la decisió d'utilitzar un derivat genèric de primer grau **BLOC*** i un de segon **BLOC**** no implicava forçosament que **BLOC*** hagués d'integrar **BLOC+** i **BLOC++**, ni que **BLOC**** hagués d'integrar **BLOC+++** i **BLOC++++**. Descartada l'opció d'un **BLOC*** que integrés **BLOC+**, **BLOC++** i **BLOC+++**, i d'un **BLOC**** equivalent a **BLOC++++**, per una excessiva especialització del primer, quedava la d'un **BLOC*** equivalent a **BLOC+**, i un **BLOC**** que integrés **BLOC++**, **BLOC+++** i **BLOC++++**, però de moment ho deixarem aquí i, si s'escau, ja reobrirem la qüestió més endavant, quan una vegada assolits els objectius principals ens poguem permetre el luxe d'optimitzar resultats.

Coneixent ja tots els paràmetres definidors de les transformacions (sistemes de referència per a les insercions, factors d'escala relatius i angle de gir en cada inserció, girs addicionals i sistemes de referència per a les definicions de **BLOC*** i **BLOC****, de forma explícita pel que fa al primer d'aquests genèrics i implícita pel que fa al segon), és el moment de decidir, com ho havíem fet dos capítols enrere per a l'encaix 2D, la millor manera d'administrar l'únic grau de llibertat que resta per tancar: l'escalat uniforme que, combinat amb el relatiu propi de

cada inserció, haurà de fer compatible el perfecte encaix 3D de cada aplicació concreta amb la mínima volada de les operacions aritmètiques, qüestió lligada a l'elecció del grau d'especificitat de **BLOC*** i **BLOC****. Ho farem recorrent de nou les tres etapes, però ara en cronologia directa, seguint el procés de creació i inserció d'aquests blocs genèrics:

1- Si, com hem fet provisionalment en descriure les etapes 3 i 2 per no complicar l'exposició, ara decidíssim també definir un **BLOC*** que en la etapa 2 poguéssim inserir amb $\mathbf{E_x = 1}$, en l'etapa 1 hauríem d'inserir el cub $\mathbf{1 \times 1 \times 1}$ de **BLOC** amb els factors d'escala $\mathbf{E_x = E_y = E_z = O\text{–}(XY')'}$, per tal que **BLOC*** fes $\mathbf{O\text{–}(XY')' \times \times O\text{–}(XY')' \times O\text{–}(XY')'}$. Però ja havíem descobert, en tractar l'encaix 2D, que un bloc dimensionat a la mida d'aquell primer cas en què es fes palesa la necessitat de comptar-hi no podia ser un genèric (utilitzable en totes les insercions de **BLOC** en què la raó $\frac{\mathbf{O\text{–}(X'')}}{\mathbf{O\text{–}(X')}}$ tingués el mateix valor) i a la vegada autosuficient (no requerir d'informació complementària, prèviament emmagatzemada, per al càlcul dels factors d'escala que calgués aplicar en les futures insercions del bloc). També havíem arribat a la conclusió que, entre les mides que podíem donar a un bloc genèric per tal que fos autosuficient, donava menys feina treballar amb el de $\frac{\mathbf{O\text{–}(X')}}{\mathbf{O\text{–}(X'')}} \times \frac{\mathbf{O\text{–}(X')}}{\mathbf{O\text{–}(X'')}}$. En conseqüència, aquí acabarem inserint **BLOC** amb els factors d'escala $\mathbf{E_x = E_y = E_z = \frac{O\text{–}(XY')'}{O\text{–}(XY')''}}$, per tal que el genèric **BLOC*** resultant tingui $\frac{\mathbf{O\text{–}(XY')'}}{\mathbf{O\text{–}(XY')''}} \times \frac{\mathbf{O\text{–}(XY')'}}{\mathbf{O\text{–}(XY')''}} \times \frac{\mathbf{O\text{–}(XY')'}}{\mathbf{O\text{–}(XY')''}}$ i en l'etapa 2 puguem inserir-lo amb uns factors tan senzills com $\mathbf{E_x = O\text{–}(XY')''}$ i $\mathbf{E_y = (Z')\text{–}(Z')''}$ (recordeu que, en NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, era $\mathbf{E_x = O\text{–}X''}$ i $\mathbf{E_y = Y\text{–}Y''}$).

2- Definit **BLOC*** en el sistema de referència "B2", per inserir-lo ens situarem en el "B". De fet, com que l'angle de gir de la inserció queda determinat per dos punts, seria igual romandre en el sistema "B2" (amb angle de gir nul) o passar al sistema "B" o a qualsevol altre que estigués centrat a **O** i tingués el mateix pla coordenat ***XY*** (amb l'angle de gir que fos): únicament caldria tenir cura de referir les coordenades del segon punt al sistema adoptat, i si preferim el "B" és per anar fent camí cap al sistema "A", que més endavant haurem de recuperar, en la línia d'uniformitzar els passos comuns a situacions diverses (quan **BLOC*** ja hagi estat creat en alguna utilització anterior de l'aplicació i ens limitem a inserir-lo per fabricar **BLOC****, no tindrem cap necessitat de recalar a "B2"). L'inserirem amb la mateixa orientació que tenia la inserció de **BLOC** quan l'hem convertida en **BLOC***, és a dir en la direcció $\mathbf{O\text{–}(XY')''}$ (el gir durà la semirecta $\mathbf{O\text{–}(XY')}$ a situar-se sobre $\mathbf{O\text{–}(XY')''}$), i així, un cop aplicats els factors d'escala proporcionals als previstos, els punts $\mathbf{(Z')'}$ i $\mathbf{(XY')'}$ del cub de referència de **BLOC*** passaran respectivament a les posicions $\mathbf{(Z')}$ i $\mathbf{(XY')}$, aquesta última ja en el pla $\mathbf{(X)\text{–}O\text{–}(Y)}$ (**X–O–Y** en l'antiga notació). Ara bé, si inseríssim aquest bloc amb els factors d'escala esmentats fa un moment, $\mathbf{E_x = O\text{–}(XY')''}$ i $\mathbf{E_y = (Z')\text{–}(Z')''}$, incorporant-hi el factor compensatori $\mathbf{E_z = O\text{–}(XY')''\ \frac{O\text{–}(X')}{O\text{–}(XY')'}}$, la inserció seria un prisma oblic de base quadrada $\mathbf{O\text{–}(X') \times O\text{–}(X')}$, és a dir que **BLOC**** ja no seria ni genèric (representatiu d'un subconjunt d'insercions de **BLOC*** en què la relació $\frac{\mathbf{O\text{–}(X'')}}{\mathbf{O\text{–}(X')}}$ tingués un mateix valor) ni tampoc autosuficient. Caldrà utilitzar els factors $\mathbf{E_x = \frac{O\text{–}(XY')''}{O\text{–}(X'')}}$, $\mathbf{E_y = \frac{(Z')\text{–}(Z')''}{O\text{–}(X'')}}$ i $\mathbf{E_z = \frac{O\text{–}(XY')''}{O\text{–}(X'')} \times \frac{O\text{–}(X')}{O\text{–}(XY')'}}$ perquè la base del

prisma oblic faci $\frac{o-(x')}{o-(x'')} \times \frac{o-(x')}{o-(x'')}$ i pugui donar lloc a un **BLOC**** que ho sigui. Aquesta qüestió n'arrossega una altra que seria bo aclarir abans de passar a l'etapa 3 i enllestir la justificació del procés: ¿per què no s'havia previst aquest problema de bon començament, i havíem adoptat en l'etapa 1 els factors $E_x = E_y = E_z = \frac{o-(xy')'}{o-(xy')'' \times o-(x'')}$ per tal d'obtenir un **BLOC*** de $\frac{o-(xy')'}{o-(xy')'' \times o-(x'')} \times$ $\times \frac{o-(xy')'}{o-(xy')'' \times o-(x'')} \times \frac{o-(xy')'}{o-(xy')'' \times o-(x'')}$, que en l'etapa 2 poguéssim inserir amb els factors $E_x = o-(xy')''$, $E_y = (z')-(z')''$ i $E_z = o-(xy')'' \frac{o-(x')}{o-(x'')}$ per constutuir el **BLOC**** esmentat? Doncs perquè llavors qui deixaria de ser autosuficient seria **BLOC***: caldria emmagatzemar el valor de $o-(x''_1)$ obtingut la primera vegada, en anar a crear **BLOC***, a fi que les futures insercions del bloc se'n poguessin servir, ja que només amb aquest valor, multiplicant pel coeficient $\frac{o-(x''_1)}{o-(x''_n)}$ els paràmetres calculats en aquelles insercions on el valor de $o-(x'')$ fos de $o-(x''_n)$, podríem convertir **BLOC*** en **BLOC**** i inserir aquest últim. ¿I, si per evitar aquest contratemps, inseríssim **BLOC** amb els factors $E_x = E_y = E_z = \frac{o-(xy')' \times o-(x')}{o-(xy')'' \times o-(x'')}$? Doncs aleshores **BLOC*** faria $\frac{o-(xy')' \times o-(x')}{o-(xy')'' \times o-(x'')} \times \frac{o-(xy')' \times o-(x')}{o-(xy')'' \times o-(x'')} \times \frac{o-(xy')' \times o-(x')}{o-(xy')'' \times o-(x'')}$, i en l'etapa 2 l'hauríem d'inserir amb els factors $E_x = \frac{o-(xy')''}{o-(x')}$, $E_y = \frac{(z')-(z')''}{o-(x')}$ i $E_z = \frac{o-(xy')''}{o-(x'')}$ per constituir el **BLOC**** esmentat, sense necessitat d'emmagatzemar cap valor per a ús de les futures insercions però amb un balanç pitjor pel que fa a economia de recursos: cada **BLOC*** concebut així seria menys genèric, perquè no seria utilitzable en totes les insercions en què $\frac{o-(xy')''}{o-(xy')'}$ tingués un valor determinat sinó únicament, dins d'aquest subconjunt, en aquelles en què $\frac{o-(x'')}{o-(x')}$ també respongués a un valor determinat. De seguida, un cop descrita l'etapa 3, tractarem precisament de com l'especificitat dels blocs genèrics **BLOC*** i **BLOC**** que hem adoptat queda reflectida en el nom d'aquests símbols. Ara, per rematar l'etapa 2, afegirem que per convertir aquesta inserció en **BLOC**** (recordeu els factors $E_x = \frac{o-(xy')''}{o-(x'')}$, $E_y = \frac{(z')-(z')''}{o-(x'')}$ i $E_z = \frac{o-(xy')''}{o-(x'')} \times \frac{o-(x')}{o-(xy')}$) cal girar el sistema de referència "A" al voltant de l'eix ***Z*** fins que el ***X*** coincideixi amb $o-(x'')$. En aquest nou sistema, que anomenarem "A2" el quadrat $\frac{o-(x')}{o-(x'')} \times \frac{o-(x')}{o-(x'')}$ quedarà inclinat un angle de cosinus $\frac{o-(x'')}{o-(x')}$.

3- Definit **BLOC**** en el sistema de referència "A2", per inserir-lo ens situarem en el "A". De fet, com que l'angle de gir de la inserció queda determinat per dos punts, seria igual romandre en el sistema "A2" (amb angle de gir nul) o passar al sistema "A" o a qualsevol altre que estigués centrat a **O** i tingués el mateix pla coordenat ***XY*** (amb l'angle de gir que fos): únicament caldria tenir cura de referir les coordenades del segon punt al sistema adoptat, i si preferim el "A" és perquè, quan el **BLOC**** que necessitem ja existeixi (igual que **BLOC***, com a producte d'anteriors aplicacions de l'eina en què treballem), no caldrà situar-se per res en "A2" però sí que haurem de recalar en "A", pas obligat per acabar l'execució deixant l'usuari en el sistema de referència vigent en el moment de començar-la. L'inserirem amb la mateixa orientació que tenia la inserció de

BLOC* quan l'hem convertida en **BLOC****, és a dir en direcció **O–(X″)** (el gir durà la semirecta **O–(X)** a situar-se sobre **O–(X″)**) i aplicarem els factors d'escala previstos, $|E_z| = E_x = O–(X'')$ i $E_y = (Y)–(Y'')$: així, els vèrtexs **(X′)** i **(Y′)** del prisma oblic de referència de **BLOC**** passaran respectivament a les posicions **(X)** i **(Y)**; també **(Z′)** (obtingut de la precedent inserció de **BLOC***) passarà a **(Z)**. Únicament quan a l'etapa 1 haguem detectat que el paral·lelogram d'acollida té una configuració "mà esquerra", és a dir, que en la transformació hi ha una simetría implícita, prendrem $E_z = -E_x$: recordeu que en el capítol precedent (des de la VERSIÓ 11+, primera en lluir el distintiu "+", precisament pel tema que esmentem) havíem decidit que **BLOC*** mantindria la configuració "mà dreta" del cub ortonormal de referència, fins i tot quan fos creat des d'un d'aquests casos (raó per la qual la inserció de **BLOC** que li donava forma es feia amb uns factors $E_x = E_y = E_z > 0$) i que la inversió no es produiria fins a la inserció final de **BLOC***, amb $E_y < 0$ (ara serà la de **BLOC****, amb $E_z < 0$); en 3D, a més, això del **BLOC*** "mà dreta" no és tan immediat, però ja en parlarem més endavant.

Considerant que el procés consistia a crear **BLOC*** a partir d'una inserció de **BLOC**, crear **BLOC**** a partir d'una inserció de **BLOC*** i inserir-lo, i que cada definició de bloc ha requerit tant especificar-ne la composició i el sistema de referència (el punt d'inserció sempre ha estat **O**) com un nom significatiu a partir del qual poguéssim conèixer l'una i l'altre, és l'hora de precisar quins són aquells trets específics de **BLOC*** i **BLOC****, comuns a cada subconjunt d'insercions en què aquests blocs siguin genèrics, i la forma de traslladar-los als seus noms, per localitzar-los i fer-ne ús (si ja existien, com a subproducte d'alguna aplicació precedent) o per crear-los (en l'aplicació actual).

Pel que fa a la definició de **BLOC***, com que **BLOC** s'ha inserit amb escalat uniforme i conserva la geometria original, el contingut queda determinat per la longitud de l'aresta del cub de referència; i l'orientació d'aquest cub en el el sistema "B2", com que el pla coordenat ***XY*** conté l'aresta ***z*** i per tant és perpendicular a la base ***xy***, queda determinada per l'angle γ_1 que forma l'aresta ***x*** amb aquest pla i l'angle β_1 que forma la base amb l'eix ***X***. Començant per aquestes dades, i tenint en compte que ens referim a l'aplicació inaugural de **BLOC*** (les aplicacions que l'usaran no passaran del procés preliminar, just per avaluar γ_1 i β_1, i poder identificar-lo):

- **BLOC** l'inserim des del sistema "A", girant-lo l'angle $\gamma = (X)–O–(X')$ que convé al cas actual, i després canviem al sistema "B" girant-lo l'angle $\gamma_2 = (X)–O–(XY')$ al voltant de ***Z*** i **±90º** al voltant de ***X***. En conseqüència, l'angle $\gamma_1 = (XY')–O–(X')$ que forma l'aresta **O–(X′)** del cub de referència amb el pla coordenat ***XY*** de "B" és $\gamma_1 = \gamma - \gamma_2$ i no dependendrà de l'aplicació actual (més endavant ja parlarem del sentit del gir **±90º**, tema relacionat amb la consecució d'un **BLOC*** "mà dreta").
- L'orientació β_1 de **BLOC** en el pla esmentat és fàcil de determinar. Com que la inserció de **BLOC** gira l'angle $\beta = (XY')–O–(XY')'$ que convé al cas actual, i canviem al sistema "B2" girant un angle $\beta_2 = (XY')–O–(XY')''$ al voltant de ***Z***, la inclinació $\beta_1 = (XY')''–O–(XY')'$ de la base del cub de referència de **BLOC**, en aquest sistema, és $\beta_1 = \beta - \beta_2$ i tampoc dependendrà de l'aplicació concreta considerada.
- Com que el cub de referència de **BLOC** és unitari, la grandària vindrà donada pels factors d'escala de la primera inserció, $E_x = E_y = E_z = \frac{O–(XY')'}{O–(XY')''} = \frac{1}{\cos\beta_1}$, i veiem que aquesta dada està lligada a l'anterior. De fet, aquesta interrelació és allò que ens permet adoptar **BLOC*** com a genèric per al subconjunt de les insercions caracteritzades per un determinat valor $\frac{O–(XY')''}{O–(XY')'} = \cos\beta_1$, autosuficient sense haver d'associar al seu nom res més que el nom de **BLOC** i els valors γ_1 i β_1: fora d'aquest cas (o bé de $E_x = E_y = E_z = 1$ o qualsevol altre constant), caldria incorporar-hi el factor únic d'escala.

Quan no hi hagi cap **BLOC*** que ens convingui, des del sistema "A" inserirem **BLOC** amb l'angle de gir γ, des del sistema "B" girarem aquesta inserció l'angle β i ens situarem en el sistema "B2" per crear-lo, amb un nom que remeti fàcilment als

valors γ_1 i β_1. De fet, no serà γ_1 el valor que incorporarem al nom de **BLOC*** sinó $45° - \gamma_1$ (en aplicació del conveni ja citat abans, no prendrem **o–(x′)** com a origen d'angles sinó la direcció **(y′)–(x′)** de la diagonal del quadrat, i d'aquesta manera no privilegiem les direccions **o–(x′)** o **o–(y′)** sinó que l'angle mitjà correspondrà a l'orientació bisectriu d'aquestes); a més, per tal de rendibilitzar els caràcters utilitzats, el rang de valors possibles (de **0°** a **180°**) es representarà amb unitats **(1- (/ 1 Q0))**èssimes (per exemple, amb una tolerància **Q0 = 0,001** els valors aniran de **0** a **999**). Quant a l'angle β_1, igual que en el capítol NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, no figurarà aquest valor en el nom de **BLOC*** sinó la part decimal de $\cos\beta_1$. La substitució de l'angle pel cosinus no suposa perdre precisió sinó just al contrari, perquè ens interessa parametritzar els resultats a partir d'aquesta funció trigonomètrica (els valors de la qual obtenim de primera mà i ens proporcionen directament el factor d'escala amb què s'insereix **BLOC**) i no a partir de l'angle. Si ara contextualitzem tot això, recordant la declaració d'intencions feta en començar el capítol, en el sentit de treballar sobre el cas general d'un **BLOC** proveït d'atributs editables, cal precisar que tot el que hem dit en relació al nom de **BLOC** s'haurà d'entendre referit als noms de **BLOC-1** i **BLOC-2**: si *B* és el nom de **BLOC** i, tal i com havíem convingut en el capítol precedent, *B_SENSE_ATRIBS* i *ATRIBS_DE_B* són els noms dels blocs substituts **BLOC-1** i **BLOC-2**, respectivament, els noms d'uns genèrics substituts de primer grau **BLOC-1*** i **BLOC-2*** podríen ser, per exemple, *B_SENSE_ATRIBS_397643* i *ATRIBS_DE_B_397643* en el cas que l'encaix del cub de referència fos un paral·lelepípede oblic (circumstància enregistrada com a **(not NORM-Z)**), tant de base romboïdal (**(not NORM-XY)**) com rectangular (**NORM-XY**). Si fos recte (**NORM-Z**) hi hauría un únic genèric de primer grau, **BLOC***, el nom del qual podria ser *B_866* (es prescindeix de γ_1 i sols queda la referència a $\cos\beta_1$), com en el plantejament 2D del capítol precedent. De moment, però, mantindrem en tots els casos la còmoda abstracció en què ens movíem, i seguirem parlant de **BLOC**, **BLOC*** i **BLOC**** com si no hi haguéssin atributs.

Pel que fa a la definició de **BLOC****, on el cub de referència s'ha transformat en un prisma oblic de base quadrada situada en el pla ***XY*** del sistema "A2", resultant d'inserir **BLOC*** en el sistema "B", el contingut queda determinat pels paràmetres d'inserció; i l'orientació d'aquest quadrat només depèn de l'angle α_1 que forma l'aresta ***x*** amb l'eix ***X*** de l'esmentat sistema "A2". Començant per aquest, i tenint en compte que ens referim a l'aplicació inaugural de **BLOC**** (les aplicacions que l'usaran no passaran del procés preliminar, just per disposar d'aquestes dades i poder localitzar-lo):

- L'orientació α_1 de la base quadrada del prisma oblic en el sistema "A" és fàcil de determinar. Com que inserim **BLOC*** amb l'angle de gir $\gamma = (x)\text{–}o\text{–}(x')$ que convé al cas actual, i després canviem al sistema "A2" girant un angle $\alpha_2 = (x)\text{–}o\text{–}(x'')$ al voltant de ***Z***, la inclinació $\alpha_1 = (x'')\text{–}o\text{–}(x')$ de l'aresta ***x*** de la base quadrada del prisma oblic de referència de **BLOC***, en aquest sistema, és $\alpha_1 = \gamma - \alpha_2$ i no dependrà de l'aplicació concreta considerada.
- En relació als paràmetres de la inserció de **BLOC***, el més immediat seria afirmar que els factors d'escala $E_x = \frac{o\text{–}(xy')''}{o\text{–}(x'')}$ i $E_y = \frac{(z')\text{–}(z')''}{o\text{–}(x'')}$ són imprescindibles (no pas $E_z = \frac{o\text{–}(xy')''}{o\text{–}(x'')} \times \frac{o\text{–}(x')}{o\text{–}(xy')}$, coincident amb el primer, tret d'un coeficient corrector), i també l'angle de gir $\beta_2 = (xy')\text{–}o\text{–}(xy')''$. Però ni de bon tros necessitem les tres dades, que estan fortament interrelacionades: com que disposem de l'angle β_1 (en realitat, de $\cos\beta_1$), n'hi haurà prou amb β_2 o, alternativament, amb la relació $\frac{E_y}{E_x} = \frac{(z')\text{–}(z')''}{o\text{–}(xy')''}$. Efectivament, si multipliquem i dividim per $o\text{–}(xy')$ el factor E_x, tindrem $E_x = \frac{o\text{–}(xy')''}{o\text{–}(x'')} = \frac{o\text{–}(xy')''}{o\text{–}(xy')} \times \frac{o\text{–}(xy')}{o\text{–}(x'')}$, i com que $o\text{–}(xy') = o\text{–}(x')$, resultarà que $E_x = \frac{o\text{–}(xy')''}{o\text{–}(xy')} \times \frac{o\text{–}(x')}{o\text{–}(x'')} = \frac{\cos\beta_2}{\cos\alpha_1}$. D'altra banda, si tornem a la Figura 8.4 de NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, considerant que **x**, **x′** i **x″** ara

són (XY'), $(XY')'$ i $(XY')''$, que Y, Y' i Y'' són (Z'), $(Z')'$ i $(Z')''$, i que la igualtat $O{-}X'' = Y'{-}Y''$ cal interpretar-la com $O{-}(XY')'' = (Z')' - (Z')''$, tindríem $E_y = E_x \frac{(Z')-(Z')''}{O-(XY')''} =$

$$= E_x \frac{(Z')-(Z')''}{(Z')'-(Z')''} = E_x \frac{(XY')-(XY')''}{(XY')'-(XY')''} = E_x \frac{\frac{(XY')-(XY')''}{O-(XY')''}}{\frac{(XY')'-(XY')''}{O-(XY')''}} = E_x \frac{\tan\beta_2}{\tan\beta_1} = \frac{\cos\beta_2}{\cos\alpha_1} \times \frac{\tan\beta_2}{\tan\beta_1},$$

és a dir, $E_y = \frac{\sin\beta_2}{\cos\alpha_1 \times \tan\beta_1}$. I pel que fa a E_z, ni tan sols depèndrà de β_2, perquè

$$E_z = E_x \frac{O-(X')}{O-(XY')'} = E_x \frac{O-(XY')}{O-(XY')'} = E_x \frac{\frac{O-(XY')}{O-(XY')''}}{\frac{O-(XY')'}{O-(XY')''}} = E_x \frac{\frac{1}{\cos\beta_2}}{\frac{1}{\cos\beta_1}} = E_x \frac{\cos\beta_1}{\cos\beta_2} = \frac{\cos\beta_2}{\cos\alpha_1} \times \frac{\cos\beta_1}{\cos\beta_2},$$

és a dir, $E_z = \frac{\cos\beta_1}{\cos\alpha_1}$. Podríem prendre $\frac{E_y}{E_x} = \frac{(Z')-(Z')''}{O-(XY')''} = \frac{\tan\beta_2}{\tan\beta_1}$, però no cal complicar-se la vida: coneixent α_1, β_1 i les seves funcions trigonomètriques, l'única dada que caldrà afegir és β_2 o alguna de les seves funcions.

Quan no hi hagi cap **BLOC**** que ens convingui, des del sistema "B" inserirem **BLOC*** amb els factors d'escala $E_x = \frac{\cos\beta_2}{\cos\alpha_1}$, $E_y = \frac{\sin\beta_2}{\cos\alpha_1 \times \tan\beta_1}$, $E_z = \frac{\cos\beta_1}{\cos\alpha_1}$ i l'angle de gir β_2, i anirem al sistema "B2" per crear-lo, amb un nom que remeti fàcilment als valors β_2 i α_1: al nom de **BLOC*** hi afegirem la part decimal de $\cos\beta_2$ i de $\cos\alpha_1$. L'elecció de la funció cosinus és conseqüent amb la de $\cos\beta_1$ per al nom de **BLOC***, ja que $\cos\alpha_1$ juga el mateix paper en la definició de **BLOC**** des del sistema "A2" que jugava $\cos\beta_1$ en la definició de **BLOC*** des del sistema "B2", i en relació a β_2 gairebé ens hi veiem forçats per exclusió de les opcions alternatives: descartada $\tan\beta_2$ pel seu il·limitat camp de variació, no hi havia cap raó per usar el valor β_2 nu, ni per criteris de precisió ni per suggerir un inexistent parentiu amb γ_1. Anàlogament a com abans havíem fet amb **BLOC***, recordant que l'existència d'aquest genèric era només virtual si de debò consideràvem blocs proveïts d'atributs N o P (igual que treballàvem amb **BLOC-1** i **BLOC-2** en comptes de **BLOC**, no hi havia un únic genèric de primer grau sinó el tàndem format per **BLOC-1*** i **BLOC-2***), ¿ara caldria abandonar temporalment llicències reduccionistes i parlar de **BLOC-1**** i **BLOC-2****? Doncs no, perquè és cert que necessitàvem diferenciar la inserció de **BLOC-2**, per tal d'explosionar-la i alliberar-ne el contingut (els atributs editables), i també diferenciar **BLOC-2*** per la mateixa raó; però una vegada disposem de la inserció de **BLOC-1*** i dels atributs lliures resultants d'explosionar **BLOC-2***, ja tindrem a mà tots els ingredients precisos per a la inserció final i només haurem d'aplegar-los en un únic bloc genèric de segon grau. Així que, justificada l'existència real de **BLOC**** i seguint amb l'exemple que havíem iniciat en relació a **BLOC***, vejam quins noms sortirien: **BLOC**** podria anomenar-se *B_397643_859866* en el cas que l'encaix del cub de referència fos un paral·lelepípede oblic de base romboïdal, perquè amb la mateixa obliqüitat però amb base rectangular (**NORM-XY**) variaria la sintaxi i el nom seria *B_397643_859* (deixant-hi la referència a $\cos\beta_2$ i prescindint de $\cos\alpha_1$, perquè l'eix d'afinitat en el pla ***XY*** del sistema "A" coincideix amb $O{-}(X)$ i $O{-}(X')$, i $\alpha_1 = 0°$); això amb el paral·lelepípede oblic, perquè si fos recte (**NORM-Z**) no hi hauría cap genèric de segon grau. És clar que, si el paral·lelepípede d'encaix fos un prisma recte de base rectangular, no necessitaríem **BLOC-1***, **BLOC-2*** ni **BLOC****, atès que n'hi hauria prou amb una inserció simple de **BLOC**.

En relació als casos d'encaix en un paral·lelepípede oblic, un podria preguntar-se quines circumstàncies haurien de concórrer per tal que dos encaixos diferents però ambdós de base romboïdal participessin d'un mateix **BLOC**** (**BLOC-1*** i **BLOC-2*** també serien compartits), o per tal que els encaixos en un de base romboïdal i en un de base rectangular, com denoten els noms de l'exemple, no només compartissin **BLOC-1*** i **BLOC-2*** sinó que els dos **BLOC****s diferents coincidissin en la seva part comuna. Basant-nos en els criteris adoptats per donar nom a cada genèric, la resposta més

expeditiva consistiria a dir que en el primer cas les insercions han de compartir els valors $999\,\frac{(45° - \gamma_1)}{180°} = 397$, $\cos\beta_1 = 0{,}643$, $\cos\beta_2 = 0{,}859$ i $\cos\alpha_1 = 0{,}866$ (hem comptat que la tolerància **Q0** val **0,001**), però que en el segon cas n'hi ha prou que comparteixin els tres primers (òbviament, no pot haver-hi coincidència en α_1 entre insercions **(not NORM-XY)** i **NORM-XY**, ja que en la primera és $0° < \alpha_1 < 90°$ mentre que en la segona és $\alpha_1 = 0°$). Malgrat tot i a risc de semblar reiteratius, provarem de presentar-ho en termes més "gràfics", reduint la forma geomètrica de **BLOC** a la del cub associat (simplificació que ens permetrà tornar a parlar de **BLOC***, en lloc de **BLOC*-1** i **BLOC-2***, i de *B_397**, en lloc de *B_SENSE_ATRIBS_397** i *ATRIBS_DE_B_397**) i considerant sempre el mateix sistema de referència (en comptes d'anar alternant canvis de ***SCP*** amb escalats i girs, prescindirem dels primers):

- Cada genèric **BLOC*** és un cub, inicialment situat sobre el sistema de referència, girat un angle γ_1 al voltant de l'eix ***Z*** i un angle β_1 al voltant de l'eix ***Y***. Si comparem **BLOC***s girats un mateix angle γ_1 però amb inclinació β_1 diferent (que donaran lloc a genèrics **BLOC**** de segon grau en què la projecció de l'aresta ***z*** sobre la base formarà amb l'aresta ***x*** d'aquesta un angle $-\gamma_1$) i que en l'exemple ($\gamma_1 = -26{,}57°$ i $999\,\frac{(45° + 26{,}57°)}{180°} = 397$) podrien respondre a la denominació *B_397**, ens adonarem que no tenen la mateixa mida sinó que l'aresta del cub fa $\frac{1}{\cos\beta_1}$.

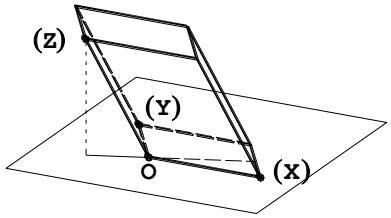
- Si "comprimim o estirem" el cub en direcció ***Z*** fins que l'angle díedre entre el pla ***XY*** i la base hagi passat de β_1 a $-\beta_2$ (parlem d'escalar en aquesta direcció, fent-ho alhora en direcció ***Y*** per mantenir la quadratura de la base), i girem el resultat (un prisma oblic, de base quadrada) l'angle β_2 al voltant de ***Y***, per tal de deixar-lo recolzat sobre el pla ***XY***, ens sortirà una geometria, a cavall entre **BLOC*** i **BLOC**** (allò que en una especulació pretèrita havíem denominat **BLOC+++**), que en l'exemple podria respondre a la denominació *B_397643_859** per tenir només definida l'orientació ***Z*** i poder girar lliurement al voltant d'aquest eix.

- Si ara considerem orientacions concretes d'aquest prisma, definides per l'angle α_1 que formi amb l'eix ***X*** l'aresta homònima de la seva base, tindrem els diversos genèrics **BLOC****: quan $\alpha_1 = 30°$ ($\cos\alpha_1 = 0{,}866$), s'anomenarà *B_397643_859866*; en el cas particular **NORM-XY**, com que $\alpha_1 = 0°$ ($\cos\alpha_1 = 1{,}000$), en lloc d'escriure *B_397643_8591000* (amb un caràcter més del permès) o *B_397643_859000*, ho deixarem en *B_397643_859*. Els **BLOC**** *B_397643_859500*, *B_397643_859866* i *B_397643_859* no diferiran únicament en l'orientació ($\alpha_1 = 60°$, **30°** i **0°**, respectivament) sinó en la grandària, com hem vist que passava amb els **BLOC***: geomètricament semblants, el costat de la base del prisma fa $\frac{1}{\cos\alpha_1}$ (**2**, **1,155** i **1**, respectivament).

- En el cas **NORM-Z** havíem qualificat **BLOC*** amb encert de genèric de primer grau, però, com que únicament depèn de l'angle α_1, la seva problemàtica té a veure amb la de **BLOC****: en l'exemple, podem considerar-lo un cas particular en què **BLOC+++** (*B_397643_859**) coincideix amb **BLOC**. Si tornem a considerar $\alpha_1 = 60°$, **30°** i **0°**, els resultats serán insercions de *B_500*, *B_866* i *B*, i l'aresta del cub fa $\frac{1}{\cos\alpha_1}$ (**2**, **1,155** i **1**, respectivament).

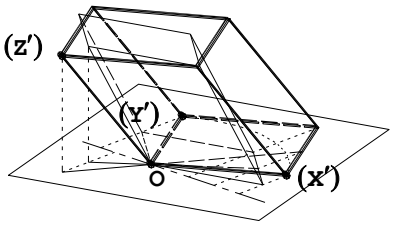
A propòsit d'això, deixem obert el tema de si de debò valia la pena de renunciar a la simplicitat conceptual d'uns genèrics **BLOC*** i **BLOC**** geomètricament congruents, és a dir, d'igual mida i no només semblants (en **BLOC**** la congruència es limitaría a base i alçada), per tal d'estalviar-nos un parell de divisions en cada inserció. Per aquest motiu, i també per facilitar gràficament el seguiment del procés, a la Figura 10.2 ens hem permès certes llibertats en relació al tema de les escales: en **E**, **BLOC** s'insereix amb uns valors $E_x = E_y = E_z$ amb els quals la base del cub de referència coincideixi amb **(X')–O–(Y')**; a **G** hem colat en el guió un escalat uniforme perquè aquesta inserció s'adapti a la secció quadrada obtinguda a **D**; la inserció de **BLOC*** la desglossem en les etapes **I** i **J** per apreciar l'escalat compensatori E_z.

A



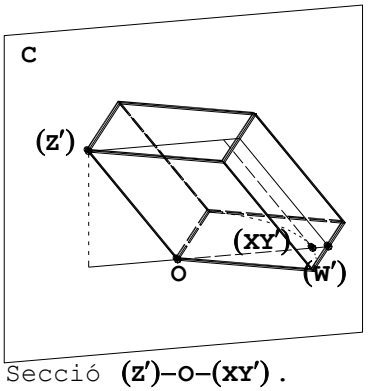
Paral·lelepípede d'encaix i adopció del sistema "A".

B



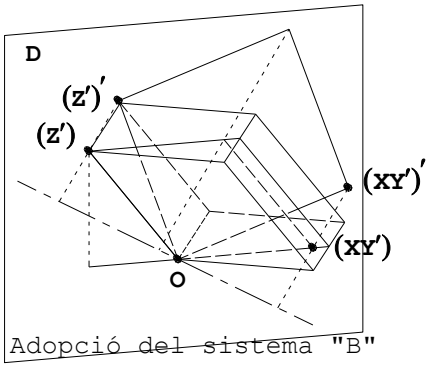
Determinació del prisma oblic de base quadrada.

C



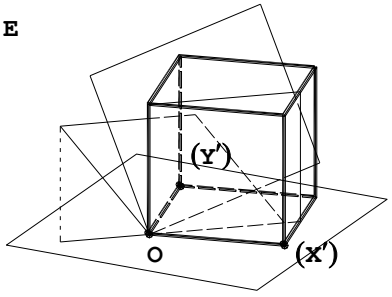
Secció **(Z′)–O–(XY′)**.

D



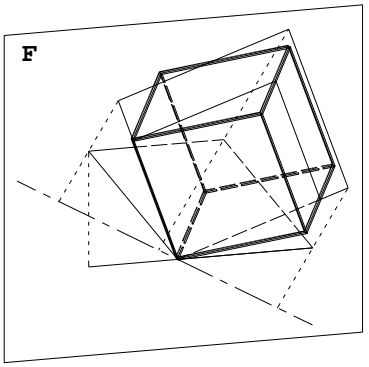
Adopció del sistema "B" i determinació del quadrat **(Z′)′–O–(XY′)′**

E



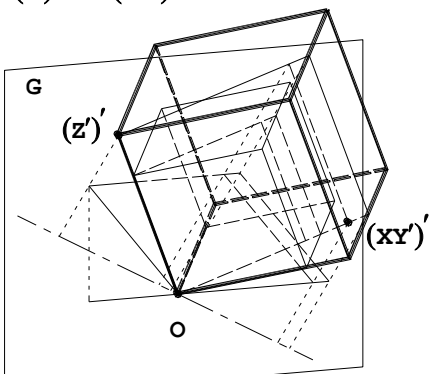
Canvi al sistema "A" i inserció de **BLOC,** girant γ i $\mathbf{E_x = E_y = E_z}$.

F



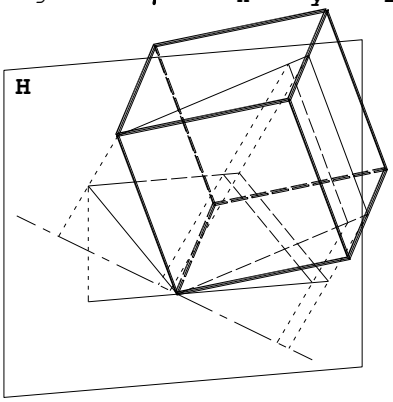
Canvi al sistema "B" i gir β.

G



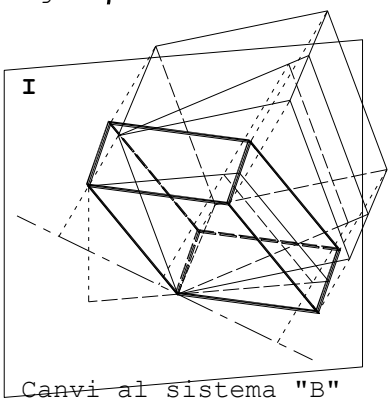
Escalat de conveniència per adaptar-se a la secció quadrada.

H



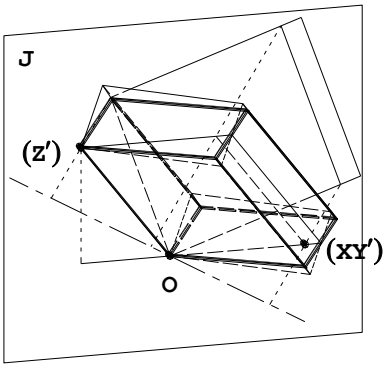
Canvi al sistema "B2" per crear **BLOC***.

I



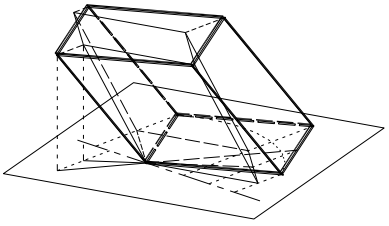
Canvi al sistema "B" i inserció de **BLOC*** girant β_2 (1ª part: $\mathbf{E_y \neq E_x}$ i $\mathbf{E_{1z} = E_x}$).

J



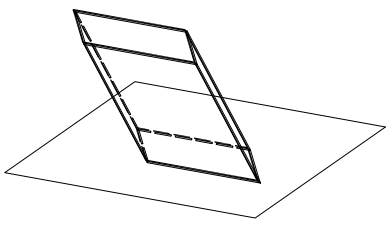
Inserció de **BLOC*** girant β_2 (2ª part: $\mathbf{E_{2z} \neq E_x}$).

K



Canvi al sistema "A2" per crear **BLOC****.

L



Canvi al sistema "A" i inserció de **BLOC**** girant α_2 i $\mathbf{|E_z| = E_x \neq E_y}$.

Figura 10.2

Ja només queda per aclarir com es resol (**+90º** o **−90º**) el gir al voltant de l'eix ***X*** que, aplicat al ***SCP*** resultant de girar el sistema "A" un angle γ_2 al voltant de ***Z***, ens ha de situar en el sistema "B".

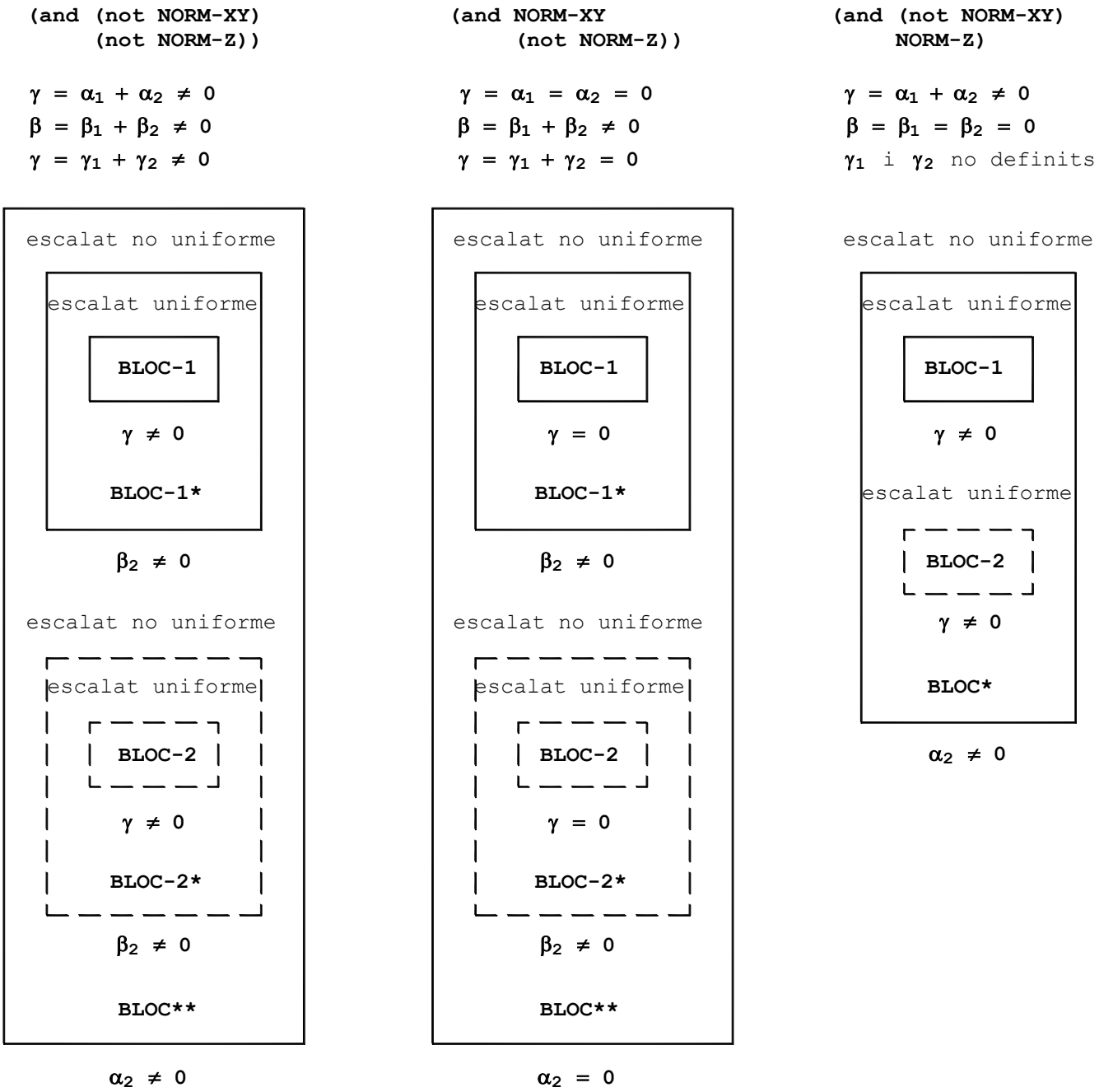
El el capítol precedent havíem decidit que la manera més elegant d'evitar efectes d'inversió de l'obliqüitat en atributs editables era assegurar-se que el genèric **BLOC*** mantingués l'orientació relativa entre elements pròpia de **BLOC**, encara que en el cas concret de l'aplicació inaugural d'aquest genèric la transformació del cub de referència de **BLOC** al paral·lelogram d'encaix dugués implícita una simetria (és a dir, el pas d'una configuració "mà dreta" a una "mà esquerra"). En 2D el canvi a "mà esquerra" es feia palès en el pla ***XY*** del ***SCP*** sota el qual s'executava l'aplicació (els canvis de ***SCP*** es reduïen a girs a l'entorn d'un eix ***Z*** que romania invariable), per ser de sentit antihorari el gir al voltant de **O** que conduïa **O−X** a alinear-se amb **O−Y** pel camí més curt (per defugir els embolics típics de tota comparació entre angles, aprofitàvem la construcció de Ritz i allò que fèiem era veure si **Y** girat **+90º** estava més a prop de **X** que girat **−90º**). I el cop de timó que donàvem en la VERSIÓ 11+ (en relació al procediment establert en la VERSIÓ 1), per tal de crear sempre un **BLOC*** "mà dreta", consistia precisament a prescindir del criteri de proximitat (gir pel camí més curt) i adoptar el gir de **−90º** (horari) que conduïa invariablement a un quadrat **X′−O−Y′** situat damunt de l'eix d'afinitat **X″−O−Y″**, es a dir, en el semiplà +***Y*** (compareu la Figura 8.5, corresponent a la VERSIÓ 1, amb la Figura 9.2, corresponent a la VERSIÓ 11+).

En 3D la construcció de Ritz l'haurem de realitzar dos cops: en el sistema "A", per situar **(X′)** i **(X″)** a partir de **(X)**, definint de passada **O−(XY′)** i el sistema "B", i en aquest segon ***SCP***, per situar $\mathbf{O-(XY')'}$ i $\mathbf{O-(XY')''}$ a partir de **O−(XY′)**. Això ens du a crear una funció **CALCULA** amb paràmetres, que s'encarregarà genèricament d'aquesta construcció. Però no és aquesta l'única diferència amb 2D. L'adopció del sistema de referència "A", que és la base de partença per a la resta de sistemes, l'aconseguirem aplicant l'opció ***3p*** de l'ordre ***SCP*** al punt d'inserció **O** i als dos primers punts de referència, **X** i **Y** (**(X)** i **(Y)**, amb la notació actual). Com que en aquest sistema és $\mathbf{X \equiv (X_x>0,0,0)}$, $\mathbf{Y \equiv (Y_x,Y_y>0,0)}$ i $\mathbf{Z \equiv (Z_x,Z_y,Z_z\neq 0)}$, quan l'encaix en el paral·lelepípede d'acollida comporti canvi a "mà esquerra" això s'acusarà en la circumstància $\mathbf{Z_z<0}$, circumstància que es manté en girar-lo l'angle γ_2 al voltant de l'eix ***Z*** i donar lloc al sistema previ a l'anomenat "B". D'altra banda, cal no oblidar que **BLOC** l'inserirem amb $\mathbf{E_x = E_y = E_z > 0}$ i, en conseqüència, serà $\mathbf{Z'_z > 0}$ [*]. La qüestió ara és: en el cas "mà esquerra", ¿hem de passar al sistema "B" girant al voltant de l'eix ***X*** **+90º** o **−90º**? Si féssim ***SCP X 90***, en definir **CALCULA** un angle $\beta_2<0$ (hem quedat que els valors calculats per aquesta funció sempre responen al supòsit "ma dreta") per donar el pas següent i anar al sistema "B2", girant β_2 al voltant de l'eix ***Z***, el paral.lelogram **(XY′)−O−(Z′)** resultaria tallat per l'eix ***X*** (l'eix d'afinitat $\mathbf{(XY')''-O-(Z')''}$), opció que ja havia quedat descartada en el capítol NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS (compareu els casos C, D, G i H de la Figura 8.3 i la Figura 8.5). En canvi, fent ***SCP X -90***, el pas a "B2" girant $\beta_2<0$ al voltant de ***Z*** donarà lloc a un eix ***X*** tal que el paral.lelogram **(XY′)−O−(Z′)** es queda en el semiplà +***Y***. ¿Així el quadrat $\mathbf{(XY')'-O-(Z')'}$, després d'haver-lo girat l'angle β ($\beta<\beta_2<0$, perquè en el sistema "A" inseríem **BLOC** amb $\mathbf{E_x = E_y = E_z > 0}$,

[*] En aquest sentit hauríem de revisar tant allò de "la transformació d'un prisma oblic de base quadrada **O−X′ × O−Y′ × O−Z′** en un paral·lelepípede oblic **O−X × O−Y × O−Z** de igual alçada (en ser coplanàries les bases inferiors, que comparteixen el vèrtex **O**, també ho seran les superiors)..." com, unes llínies més avall, allò de "remuntant-nos ara al pla de les bases superiors, veurem com la recta **Z−Z′** (**O−Z** no és perpendicular a la base) és paral·lela a **X−X′** i a **Y−Y′**", afirmat a l'inici de l'exposició **3 → 2 → 1**, perquè als efectes de poder sentenciar que "si anomenem **Z″** la intersecció de **Z−Z′** amb el pla normal a les bases i que passa per l'eix d'afinitat **X″−O−Y″**, també s'acomplirà que $\frac{Z''-Z}{Z''-Z'} = \frac{X''-X}{X''-X'} = \frac{Y''-Y}{Y''-Y'}$", com fèiem tot seguit, n'hi havia prou a considerar que en aquell context **Z** i **Z′** representaven la projecció d'aquests punts sobre el pla coordenat ***XY***, i **Z″** la intersecció de **Z−Z′** amb l'eix d'afinitat **X″−O−Y″**.

de manera que l'angle $\beta_1 = \beta - \beta_2$, el cosinus del qual és un dels identificadors de **BLOC***, també serà $\beta_1 < 0$), se situarà en el semiplà $-Y$, en franca contradicció amb la fita que havíem establert des de la VERSIÓ 11+? No, perquè aquest ***SCP X -90*** té lloc en allò que a l'inici del capítol anomenàvem procés preliminar $3 \rightarrow 2 \rightarrow 1$, únicament per al càlcul dels paràmetres d'inserció de **BLOC** i definició de **BLOC***. Quan de debó anem a realitzar aquestes operacions (procés d'execució $1 \rightarrow 2 \rightarrow 3$), el trànsit des del sistema "A", girat un angle γ_2 al voltant de Z, al sistema "B" el farem mitjançant ***SCP X 90***, de manera que la situació en el pla XY s'invertirà: el quadrat $(XY')'-O-(Z')'$ (el cub de referència de la inserció de **BLOC**) se situarà en el semiplà $+Y$, igual que en els casos amb l'encaix "mà dreta", i el paral·lelogram $(XY')-O-(Z')$ en el semiplà $-Y$, havent girat l'angle β (ara serà $\beta > \beta_2 > 0$, de manera que l'angle $\beta_1 = \beta - \beta_2$ també serà $\beta_1 > 0$). I un cop disposem de **BLOC****, la inserció final des del sistema "A", amb $E_z < 0$, ens garantirà la inversió a "mà esquerra", fent coincidir l'homòleg del punt $(0,0,1)$ del cub de referència (en **BLOC****) amb el punt **Z** del paral·lelepíped d'acollida.

Figura 10.3



Amb l'aclariment relatiu al gir al voltant de ***X*** donem per acabada l'exposició, tot i que abans de passar a les qüestions més especificament de programació seria bo recapitular. Ho farem seguint el fil del cas general (**(and NORM-XY (not NORM-Z))** i **(and (not NORM-XY) NORM-Z)** els hem esquematitzat a la Figura 10.3, i sabem que el cas **(and NORM-XY NORM-Z)** es resol amb una inserció simple) i ignorant els atributs per últim cop, per simplificar i poder escriure **BLOC*** en lloc de **BLOC-1***/**BLOC-2***:

Procés preliminar:

- ***SCP*** per situar-nos en el sistema "A": el pla ***XY*** és **(X)–O–(Y)** i l'eix ***X*** és **O–(X)**.
- A partir de **(X)**, **(Y)**, **O–(Y)** i del caràcter agut/obtús de l'angle **(X)–O–(Y)**, la funció **CALCULA** troba **(X′)**, **(X″)**, **O–(X′)**, **O–(X″)** i **(Y″)**.
- A partir d'aquests valors i de **(Z)**, trobarem **(Z″)**, **(Z′)**, i **(XY′)**.
- ***SCP*** per girar l'angle γ_2 **= (X)–O–(XY′)** al voltant de ***Z***: el mateix pla ***XY*** i l'eix ***X*** és **O–(XY′)**.
- ***SCP*** per situar-nos en el sistema "B" (o a l'invers) girant **±90º** al voltant de ***X***: el pla ***XY*** és **(XY′)–O–(Z′)** i l'eix ***X*** és **O–(XY′)**.
- A partir de **(XY′)**, **(Z′)**, **O–(Z′)** i del caràcter agut/obtús de l'angle **(XY′)–O–(Z′)**, la funció **CALCULA** troba $(XY')'$, $(XY')''$, $O–(XY')'$, $O–(XY')''$ i $(Z')''$.
- ***SCP PRev*** dues vegades, per tornar al sistema "A".
- Amb el nom **BLOC** i els valors γ_1 **= (XY′)–O–(X′)**, $\beta_1 = (XY')''–O–(XY')'$, β_2 **= (XY′)–O–**$(XY')''$ i $\alpha_1 = (X')''–O–(X')'$, muntem els noms **BLOC*** i **BLOC****.

Procés executiu:

- Si **BLOC*** no existeix:
  - ***INSERT*** per inserir **BLOC** amb $E_x = E_y = E_z = \dfrac{O–(XY')'}{O–(XY')''}$ i angle de gir **γ = (X)–O–(X′)**.
  - ***SCP*** per girar l'angle γ_2 al voltant de ***Z***.
  - ***SCP*** per situar-nos en el sistema "B" girant **+90º** al voltant de ***X***.
  - ***GIRA*** la inserció de **BLOC** un angle $\beta = (XY')–O–(XY')'$.
  - ***SCP*** per situar-nos en el sistema "B2" girant $\beta_2 = (XY')–O–(XY')''$ al voltant de ***Z***: el mateix pla ***XY*** i l'eix ***X*** és $O–(XY')''$.
  - ***BLOQUE*** per definir el genèric de primer grau **BLOC***.
  - ***SCP PRev***, per tornar al sistema "B".
- Si **BLOC**** no existeix:
  - Si **BLOC*** ja existia en iniciar el procés:
    - ***SCP*** per girar l'angle γ_2 al voltant de ***Z***.
    - ***SCP*** per situar-nos en el sistema "B" girant **+90º** al voltant de ***X***.
  - ***INSERT*** per inserir **BLOC*** amb $E_x = \dfrac{O–(XY')''}{O–(X'')}$, $E_y = \dfrac{(Z')–(Z')''}{O–(X'')}$, $E_z = \dfrac{O–(XY')''}{O–(X'')} \times \dfrac{O–(X')}{O–(XY')'}$ i angle de gir $\beta_2 = (XY')–O–(XY')''$.
  - ***SCP PRev*** dues vegades, per tornar al sistema "A".
  - ***SCP*** per situar-nos en el sistema "A2" girant $\alpha_2 = (X)–O–(X'')$ al voltant de ***Z***: el mateix pla ***XY*** i l'eix ***X*** és **O–(X″)**.
  - ***BLOQUE*** per definir el genèric de segon grau **BLOC****.
  - ***SCP PRev***, per tornar al sistema "A".
- ***INSERT*** per inserir **BLOC**** amb $E_x = O–(X'')$, $E_y = (Y)–(Y'')$, $E_z = \pm\, O–(X'')$ i angle de gir $\alpha_2 = (X)–O–(X'')$.

Justificada l'estratègia que se segueix per estendre a 3D el problema 2D resolt en el capítol precedent, ha arribat l'hora d'abordar-ne la implementació en AutoLISP, però com que ja ens hem esplaiat prou descrivint el procés geomètric, evitarem ser reiteratius i, en lloc de connectar amb l'última versió i progressar pas a pas com fins ara, presentarem directament el nou codi, sota la denominació de VERSIÓ 16+: les diferències més significatives respecte a les anteriors, que es donen sobretot a **C:INSERTOK** i a **INSERT***, gairebé surten soles com a adaptació al nou plantejament geomètric i entenem que no precisen d'explicacions addicionals; pel que fa als aspectes secundaris, els comentem tot seguit, abans de l'esmentada presentació.

Per tal d'evitar-li a l'usuari preguntes prèvies tan enutjoses com *¿Bastará con situar los extremos de segmentos +X y +Y (adaptar un paralelógramo ortogonal a uno oblicuo) o también hay que situar +Z (adaptar un paralelepípedo ortogonal a uno oblicuo)?*, hem cregut convenient que, en comptes d'haver-hi una única nova ordre, ***INSERTOK***, identificada amb la funció AutoLISP **C:INSERTOK** sense arguments, l'usuari en tingui dues al seu abast, amb els prou expressius noms ***INS2D*** i ***INS3D***, i que les funcions **C:INS2D** i **C:INS3D** es limitin a fer de pont, diversificant l'accés a una funció **INSERTOK** que ara comptarà amb l'argument booleà **2D**:

```
(defun C:INS2D () (INSERTOK T))
(defun C:INS3D () (INSERTOK ()))
```

Més enllà d'aquesta qüestió formal, convé remarcar, per damunt de les aparences (el retorn a la llista **WWAA** de valors per als atributs Normals, verificables o no, no sol·licitats en temps real sinó prèviament, a **VAL-ATRIBS**), que la VERSIÓ 16+ no entronca amb la VERSIÓ 14+, saltant des d'allà de 2D a 3D, sinó amb una VERSIÓ 15+ modificada precisament en allò que a primera vista la diferencia de la precedent: poder prescindir de la llista **WWAA** i assignar valors als atributs des de l'Editor de Dibuix. Diem això perquè l'especificitat fonamental de la VERSIÓ 14+ residia a la manera d'afrontar els desplaçaments soferts pels atributs no justificats sobre la línia base (editar-los després de l'última inserció), mentre que la VERSIÓ 15+ es caracteritzava per eludir el problema canviant en el substitut **BLOC-2** el tipus de justificació d'aquests atributs.

A la VERSIÓ 14+, allò que ens obligava a construir **WWAA** era la impossibilitat que l'últim ***INSERT*** fos l'última ordre executada des de **C:INSERTOK**, per la necessitat d'editar aquesta darrera inserció (i en aquest sentit, tant s'hi valia si ho fèiem amb l'ordre ***-ATREDIT*** com amb les funcions **entmod** i **entupd**), és a dir que el recurs a la llista esmentada només era conseqüència d'aquesta necessitat. Paradoxalment, havent optat ara per donar continuïtat a la VERSIÓ 15+, prescindint doncs d'edició d'atributs i semblant que ja res no ens hauria d'impedir seguir assignant valors als atributs N en temps real, la irrupció en escena d'unes inoportunes ordres ***SCP*** que emmarquen l'accés a ***INSERT*** (executada un sol cop, si es dóna **NORM-XY** i **NORM-Z**, o reiteradament, en cas contrari, dintre de **INSERT***) constitueix el nou obstacle. A diferència del plantejament 2D, on donàvem per suposat que el pla ***XY*** del sistema de referència vigent en començar l'execució coincidia amb el del paral·lelogram d'encaix **(X)–O–(Y)**, aquí ens ho volem assegurar d'entrada: mitjançant l'expressió **(command ... "SCP" "3" O X Z-XY)** on la variable **Z-XY** representa provisionalment **Y**, implantem el sistema que abans anomenàvem "A" i, després de l'única o última ordre ***INSERT***, l'expressió **(command "SCP" "PR" ...)** ens retorna al sistema de referència previ a l'execució de ***INS2D*** o ***INS3D*** (si a continuació també restaurem els valors de ***INSNAME***, ***ATTDIA***, ***OSMODE*** i ***CMDECHO*** sota la cobertura de **command**, en comptes de fer-ho amb **setvar**, és per aprofitar aquest paraigua, opció legítima perquè el mal ja està fet). Tanmateix, quan en el capítol següent ho fem "més difícil encara", contemplant blocs amb atributs col·locats lliurement, que hauran de ser tractats al marge dels que considerem aquí (atributs P i/o N que sempre se situen en el pla ***XY*** del sistema de referència propi de **BLOC** i que estan adscrits a **BLOC-2**), raó per la qual caldrà reordenar el conjunt, l'aparent retrocés d'ara afavorirà el procés: com que les sol·licituds de valor s'hauran de succeir en el mateix ordre que en el cas d'una inserció simple de **BLOC**, els valors només es podran usar com arguments de ***INSERT*** en diferit, després d'haver reagrupat els atributs convencionals sobre **BLOC-2**, d'una banda, i els col·locats lliurement sobre blocs *ad hoc*, de l'altra.

Contràriament a la VERSIÓ 16+ que anirem perfilant en el curs del present capítol, on l'assignació de valor als atributs (**VAL-ATRIBS** i **VATR**) es produeix abans que s'activi el dispositiu ***DESHACER Inicio/Fin*** (amb el qual perseguim que ***INS2D***/***INS3D*** sigui tractada com una ordre ordinària als efectes d'usar ***DESHACER*** o ***H*** després de la seva execució, com recordareu del capítol OPTIMITZACIÓ DELS RECURSOS EXISTENTS:

2 EXEMPLES), i això garanteix que cap incidència de mecanografiat no repercutirà negativament sobre l'eficàcia del referit dispositiu en introduir aquests valors per mitjà de **getstring** (l'instrument que **VATR** usa), en la VERSIÓ 18+ i posteriors, ja a l'esmentat capítol següent, no tindrem tanta sort. De fet, no serà tant la presència de l'expressió **(command ... "SCP" "3" O X Z-XY)** com el fet que vagi per davant de l'assignació de valor als atributs (funcions **VAL-ATRIBS-1**, **VAL-ATRIBS-2** i **VATR**) allò que ens farà la guitza i ens obligarà a complicar una mica les coses. De moment, aquí ens en hem pogut sortir com el lector aplicat ja haurà descobert: situant **VAL-ATRIBS** fora de **INSERT***, abans de la referida expressió; si l'execució ara està condicionada a **(= (logand (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) 2)** i no a l'existència de la llista **OO-2**, com a la VERSIÓ 15+, és perquè el trasllat ha comportat duplicar el procés de cerca d'atributs a partir del primer component **(cdr (assoc -2 (tblsearch "BLOCK" BLOC)))** de **BLOC**, en haver-nos hagut d'avançar a la feina de **SEGR-ATRIBS** (constitució de les llistes **OO-1** i **OO-2** per donar vida als substituts **BLOC-1** i **BLOC-2**).

Referint-nos ja a bocins de codi concrets, una primera constatació pràctica és que al procés d'inserir **BLOC-1** i **BLOC-2** amb els mateixos paràmetres, explosionar la segona inserció i construir amb el productes resultants (la inserció de **BLOC-1** i els atributs alliberats en l'explosió de **BLOC-2**) un genèric derivat de primer grau **BLOC***, li surt una variant que ens condueix irremeiablement a un procés similar: que en comptes d'aplegar-ho tot en un únic genèric **BLOC*** (això només ho farem si es dóna la condició **NORM-Z**, com quan ens movíem en 2D), de la inserció de **BLOC-1** en farem **BLOC-1*** i del conjunt d'atributs en farem **BLOC-2*** per, un cop calculada la nova col·lecció de paràmetres, utilitzar-la en la inserció de **BLOC-1*** i **BLOC-2*** i Sant Tornem-hi (explosió de **BLOC-2*** i construcció d'un genèric derivat de segon grau **BLOC****, que en el codi no apareixerà amb aquest nom sinó reutilitzant **BLOC***). Amb aquest panorama, no serà cap poca-soltada mirar d'evitar allò que de la VERSIÓ 13+ ençà encara havíem considerat admissible: la repetició del fragment de codi

```
(command "INSERT" BLOC-1 ...)
(setq SS (ssadd (entlast)))
(if OO-2
    (progn
      (command "ATTREQ" 0
               "INSERT" BLOC-2 ...
               "ATTREQ" (if ATREQ 1 0)
               "DESCOMP" (entlast))
      (setq SA (ssget "P") K -1)
      (while (setq E (ssname SA (setq K (1+ K))))
             (ssadd E SS))))
(command ... "BLOQUE" ... SS "")
```

en les funcions **REDEF-BLOC*S** i **INSERT***, que podia justificar-se pel fet que en la primera es localitzava en un únic bucle **while**, que començàvem des de **SEGR-ATRIBS** per evitar un inútil ball de variables (entre elles ***ATTREQ***, que quedava fora) si acabàvem sense cap genèric per actualitzar, i només en el cas de trobar-ne algun continuàvem des de **REDEF-BLOC*S**, i que en la segona també sortia una sola vegada. Ara, en multiplicar-se les aparicions (tres vegades en **INSERT***, corresponents al tàndem **BLOC-1***/**BLOC-2***, a **BLOC*** i a **BLOC****, i dues en **REDEF-BLOC*S**, on el primer bucle **while** es consagra a la cerca i actualització dels **BLOC****s mentre el segon tant serveix per als tàndems **BLOC-1***/**BLOC-2*** com per als **BLOC***s), serà inexcusable no definir amb aquest fragment de codi (n'exclourem l'última línia, més versàtil) una funció que anomenarem **INSPARCIAL**. Per tal de no incorporar-hi més arguments, utilitzarem les variables globals **BLOC-1** i **BLOC-2**, de manera que algun cop haurem d'efectuar transferències de valor des de (o cap a) **BLOC-1*** i **BLOC-2***:

```
(defun INSPARCIAL (EX EY EZ ANG)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast)))
  (if OO-2
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast))
        (setq SA (ssget "P") K -1)
        (while (setq E (ssname SA (setq K (1+ K))))
               (ssadd E SS)))))
```

Pel que fa a l'exclusió de la línia **(command ... "BLOQUE" ... SS "")**, on els punts suspensius tapaven diferències entre **INSERT*** i **REDEF-BLOC*S**, aquestes diferències s'han ampliat en la mesura que en les dues funcions s'ha multiplicat la presència d'aquell fragment de codi, i es detecten fàcilment en les versions que veurem ara mateix: només cal localitzar-hi les crides a **INSPARCIAL** i analitzar-ne el context. Començant per la primera, i amb l'única excepció del cas **NORM-Z** on conflueixen els plantejaments 3D (***INS3D***) i 2D (***INS2D*** o ***INSERTOK*** en les versions prèvies), amb un genèric **BLOC*** comú, la transcendència dels canvis de ***SCP*** que ara es donen entre la inserció de cada tàndem (amb la descomposició del segon bloc) i la conversió del producte resultant en bloc derivat (incloent-hi ***GIRA***, en el cas de la creació de **BLOC-1***/**BLOC-2*** a partir de **BLOC-1**/**BLOC-2**), comparada amb els canvis irrellevants de ***SCP*** que es produïen en 2D en anàloga situació (girs al voltant de l'eix ***Z***, que no comportaven canvi de pla coordenat ***XY***), revela el fet que els genèrics 3D estan definits des de sistemes de referència ben diferents dels que havien propiciat la inserció dels seus components. En la funció **REDEF-BLOC*S** hi veurem reflectides les mateixes qüestions: en 2D (parlem de VERSIÓ 13+ a VERSIÓ 15+, i també de l'actual VERSIÓ 16+ pel que fa als genèrics sorgits de la situació **NORM-Z**) podíem i podem seguir inserint els components actualitzats del tàndem **BLOC-1**/**BLOC-2** i redefinir el genèric **BLOC*** en el mateix ***SCP***, però quan vulguem redefinir genèrics 3D haurem de fer **(command "SCP" "EZ" 0 (cdr (assoc 210 LE)))** abans de recórrer a **INSPARCIAL** i **(command "SCP" "PR")** després; el codi **210** de les insercions de **BLOC-1** i **BLOC-1*** representa l'orientació ***Z*** del sistema amb què els havíem inserit (relativa al que hi havia en el moment de crear el genèric), igual al de les insercions de **BLOC-2** i **BLOC-2*** (una vegada alliberats els atributs per l'explosió d'aquestes últimes, el seu codi **210** ja no serà el mateix perquè l'orientació es referirà al ***SCU****niversal*).

Una altra de les complicacions de **REDEF-BLOC*S** és que, quan haguem localitzat un **BLOC*** o un **BLOC**** que calgui redefinir, farem **(command "BLOQUE" BLOC* "S" 0 SS "")** (recordem que **BLOC**** també apareix en el codi sota el nom **BLOC***), on l'argument **"S"** representa manifestar la conformitat amb el missatge *El bloque ... ya existe. ¿Desea volver a definirlo? [Sí/No] <N>:*, però que en no totes les situacions és tan clara la pertinència d'aquest argument (sigui **"S"** o **"N"**, això tant se val): quan localitzem un **BLOC-1*** també farem **(command "BLOQUE" BLOC-1* "S" 0 ...)**, però ¿què farem amb **BLOC-2***? Recordem que el dibuix no té cap inserció de **BLOC-2** ni de **BLOC-2***, perquè totes han estat explosionades immediatament, que aquests blocs han pogut ser víctimes fàcils d'ordres ***LIMPIA*** a càrrec d'usuaris inexperts, i que la possibilitat de desempallegar-se d'aquests blocs substituts era justament allò que en el capítol precedent ens havia fet concebre la idea de usar aquest recurs per desencadenar una actualització de tots els substituts i genèrics a partir de la redefinició del **BLOC** original. Per tal d'afrontar la incertesa podríem caure en la tentació d'utilitzar de forma extensiva la substitució de **command** per la llista sense avaluar **(eval (append '(command ...)))**, com mostrem en la versió següent, on per entendre el joc amb els perfils de **wcmatch** cal tenir en compte el valor de **M** i la seva intervenció en la composició de **N** (veieu-ho a l'inici de **INSERTOK**, cap al final del capítol), ja que difereix de la VERSIÓ 15+:

```
(defun REDEF-BLOC*S (/ LB)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (setq J (1+ (strlen BLOC-1)))
  (tblnext "BLOCK" T)
  (while (setq LB (tblnext "BLOCK"))
         (if (wcmatch (setq BLOC-1* (cdr (assoc 2 LB)))
                      (strcat BLOC-1 M (substr M 2)))
             (progn
               (setq E (cdr (assoc -2 LB)) LE (entget E)
                     K (cdr (assoc 41 LE))
                     BLOC-2* (strcat BLOC-2 (substr BLOC-1* J)))
               (command "SCP" "EZ" 0 (cdr (assoc 210 LE)))
               (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
               (eval (append
                       '(command "SCP" "PR"
                                 "BLOQUE" BLOC-1* "S" 0 (setq E (ssname SS 0)) ""
                                 "BLOQUE" BLOC-2*)
                                          (if (tblsearch "BLOCK" BLOC-2*) '("S"))
                                          '(0 (ssdel E SS) ""))))))
```

```
  (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
  (tblnext "BLOCK" T)
  (while (setq LB (tblnext "BLOCK"))
         (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat N "*"))
             (progn
               (setq E (cdr (assoc -2 LB)) LE (entget E)
                     BLOC-1 (cdr (assoc 2 LE))
                     LB (substr BLOC-1 J)
                     BLOC-2 (strcat BLOC-2* LB))
               (if (> LB "") (command "SCP" "EZ" O (cdr (assoc 210 LE))))
               (INSPARCIAL (cdr (assoc 41 LE)) (cdr (assoc 42 LE))
                           (cdr (assoc 43 LE)) (/ (* (cdr (assoc 50 LE)) 180) PI))
               (if (> LB "") (command "SCP" "PR"))
               (command "BLOQUE" BLOC* "S" O SS ""))))
  (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2*)
  (setvar "CLAYER" CAPA))
```

Pel que fa a **INSERT*** també podríem tirar de veta i seguir utilitzant el dispositiu **(eval (append '(command ...)))**, perquè de situacions d'incertesa per la possible existència prèvia de blocs amb el nom d'aquell que volem crear encara n'hi ha més: en concret, a l'hora de definir **BLOC-1*** i **BLOC-2***, crearem aquests genèrics només si **(not (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))**; dels tres casos que tenen cabuda en l'enunciat precedent, i que són

1- **(and (not (tblsearch "BLOCK" BLOC-1*)) (not (tblsearch "BLOCK" BLOC-2*)))**
2- **(and (tblsearch "BLOCK" BLOC-1*) (not (tblsearch "BLOCK" BLOC-2*)))**
3- **(and (not (tblsearch "BLOCK" BLOC-1*)) (tblsearch "BLOCK" BLOC-2*))**,

els dos últims comporten redefinició i són ben plausibles. Pot ser que **BLOC-1*** existeixi però **BLOC-2*** no, per haver-lo destruït directament amb **LIMPIA** sense fer el mateix amb **BLOC-2**, i també que **BLOC-2*** existeixi però **BLOC-1*** no, per haver esborrat totes les insercions dels genèrics **BLOC**** que depenen de **BLOC-1*** i haver eliminat després amb ***LIMPIA*** tots dos blocs, sense afectar els **BLOC-2***s ni **BLOC-2**: com que la manera de desfermar la redefinició dels substituts **BLOC-1** i **BLOC-2** i de tots els genèrics existents és precisament desempallegar-se d'aquest segon bloc, **SEGR-ATRIBS** no haurà actuat actualitzant aquest tàndem ni menys encara executant **REDEF-BLOC*S** per actualitzar o restaurar genèrics. La funció quedaria així:

```
(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** Z-XY* XY XY* XY** OXY* OXY** SA SS
                  OO-1 OO-2 BLOC-1 BLOC-2 BLOC* BLOC-1* BLOC-2* BL*EX)
  (setq BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'OX* 'OX** 'Y**)
        (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
              A (inters Z-XY (polar Z-XY W 1) O X** ())
              Z-XY* (polar A (angle A Z-XY)
                           (/ (* (distance A Z-XY) (distance X* X**))
                              (distance X X**))))))
  (if NORM-Z
      (setq BLOC* (strcat BLOC "_"))
      (progn
        (setq X (inters X* Y* O Z-XY* ())
              X (if X X Z-XY*)
              Z-XY (- (angle O X) (angle Y* X*))
              Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
              Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
              Z (trans (list (car Z-XY*) (cadr Z-XY*) (last Z)) 1 0))
        (command "SCP" "Z" O X
                 "SCP" "X" (* I 90))
        (setq XY (list OX* 0 0) Z (trans Z 0 1))
        (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'OXY* 'OXY** 'Z**)
```

```
      (setq K (strcat "_" (itoa (fix (* (- 1 Q0) (/ Z-XY PI Q0))))
                          (itoa (fix (/ OXY** OXY* Q0))))
            BLOC* (strcat BLOC K)
            BLOC-1* (strcat BLOC-1 K) BLOC-2* (strcat BLOC-2 K)
            BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
      (command "SCP" "PR" "SCP" "PR")
      (if (not BL*EX)
          (progn
            (INSPARCIAL (setq K (/ OXY* OXY**)) K K X*)
            (eval (append
                    '(command "SCP" "Z" O X
                              "SCP" "X" 90
                              "GIRA" SS "" O XY*
                              "SCP" "Z" O XY**
                              "BLOQUE" BLOC-1*)
                                       (if (tblsearch "BLOCK" BLOC-1*) '("S"))
                                       '(O (setq E (ssname SS 0)) ""
                              "BLOQUE" BLOC-2*)
                                       (if (tblsearch "BLOCK" BLOC-2*) '("S"))
                                       '(O (ssdel E SS) ""
                              "SCP" "PR")))))
      (setq B BL*EX BLOC BLOC* BLOC-1 BLOC-1* BLOC-2 BLOC-2*
            BLOC* (strcat BLOC "_" (itoa (fix (/ OXY** OX* Q0)))))))
(if (not NORM-XY) (setq BLOC* (strcat BLOC* (itoa (fix (/ OX** OX* Q0))))))
(if (or NORM-Z BL*EX) (setq BL*EX (tblsearch "BLOCK" BLOC*)))
(if (not BL*EX)
    (progn
      (if NORM-Z
          (INSPARCIAL (setq K (/ OX* OX**)) K K X*)
          (progn
            (if B (command "SCP" "Z" O X
                           "SCP" "X" 90))
            (INSPARCIAL (setq K (/ 1 OX**) E (* OXY** K))
                        (* (distance Z Z**) K) (* E (/ OX* OXY*)) XY**)
            (command "SCP" "PR" "SCP" "PR")))
      (command "SCP" "Z" O X**
               "BLOQUE" BLOC* O SS ""
               "SCP" "PR")))
(command "CELWEIGHT" GLIN
         "CELTYPE" TLIN
         "CECOLOR" COL
         "CLAYER" CAPA)
(eval (append '(command "INSERT" BLOC* O "XYZ" OX** (distance Y Y**)
                        (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1) OX** I) X**)
              (if ATREQ WWAA))))
```

El recurs a les llistes sense avaluar només està prou justificat en l'expressió de les 3 últimes línies, per si s'escau donar cabuda als arguments que han d'assignar als atributs N de **BLOC*** els valors recollits prèviament per la funció **VAL-ATRIBS.** Pel que fa a l'eventual preexistència d'alguns blocs **BLOC-1*** (a **INSERT***) i **BLOC-2*** (a **REDEF-BLOC*S** i a **INSERT***), no cal complicar-se la vida amb aquests dispositius (que ultra ser més llargs retarden l'execució en la mesura que, si volem convertir l'arxiu .LSP a format .FAS o .VLX, no podran compilar-se i hauran de ser avaluats en temps real): canviant a **2** la variable de sistema ***EXPERT*** n'hi haurà prou per deslliurar-nos de la penosa obligació d'autoritzar la redefinició d'aquests blocs; i quan d'aquí a un moment tornem a presentar les funcions **REDEF-BLOC*S** i **INSERT***, ja incorporaran aquesta simplificació.

Una incidència nova, que ara no només se'ns pot presentar explosionant insercions obliqües amb ***DESCOMPOK*** sinó realitzant-les amb ***INSERTOK***, és la minva d'alçada que experimenten els atributs definits amb caràcters oblics. Recordareu que en el capítol precedent, mentre preparàvem la VERSIÓ 14+, havíem descobert dos "efectes col·laterals": l'un afectava els atributs no justificats sobre la línia base, en les circumstàncies que explicàvem, consistia en un desplaçament en la direcció d'escriptura i en la VERSIÓ 15+ vèiem que la manera més pràctica d'evitar-lo era substituir aquests atributs a **BLOC-2** per uns altres que estiguéssin justificats sobre la línia base; l'altre afectava els atributs escrits amb caràcters oblics, quan explosionàvem una inserció del bloc portador realitzada amb factors d'escala

no uniformes, consistia en un aixafament dels caràcters en direcció transversal i ja a la VERSIÓ 14+ trobàvem la manera de neutralitzar-lo aplicant la deformació inversa. Doncs bé, si en 2D el segon d'aquests efectes ja s'apreciava en ***DESCOMPOK*** (on arribàvem a explosionar insercions obliqües del genèric de primer grau **BLOC***) però no encara en ***INSERTOK*** (on obteníem els atributs P i N de **BLOC*** explosionant una inserció $\mathbf{E_x = E_y}$ de **BLOC-2**), en 3D (on, quan **(not NORM-Z)**, la mateixa funció obté els atributs P i N de **BLOC**** explosionant insercions obliqües del genèric de primer grau **BLOC-2***) sí que es produeix l'aixafament i caldrà neutralitzar-lo. Però no només en la penúltima inserció **(not NORM-Z)** realitzada des de **INSERT*** pot passar això i caldrà intervenir, sinó també quan redefinim des de **REDEF-BLOC*S** els genèrics **BLOC**** per donar entrada als canvis introduïts a **BLOC**, després d'haver-los repercutit en **BLOC-1** i **BLOC-2**, i tot seguit en **BLOC-1*** i **BLOC-2*** (tanmateix, no passarà quan redefinim els genèrics **BLOC*** apareguts en circumstàncies **NORM-Z**).

Sortosament no caldrà repetir el dispositiu neutralitzador dintre de **C:INSERTOK** (de **C:DESCOMP** ja ens en ocuparem a continuació), perquè la situació crítica es produeix dintre de la funció I**NSPARCIAL**, accedida des de **REDEF-BLOC*S** i **INSERT***, i aquí és on l'integrarem. Només caldrà afegir-hi un cinquè argument, **COMPROBLIC**, per marcar aquelles intervencions on haguem vist que pot produir-se l'aixafament, comprovar la obliqüitat dels caràcters i, si cal, restituir-los l'altura original.

```
(defun INSPARCIAL (EX EY EZ ANG COMPROBLIC)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast)))
  (if OO-2
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast))
        (setq SA (ssget "P") K -1
              E* (if COMPROBLIC (cdr (assoc -2 (tblsearch "BLOCK" BLOC-2)))))
        (while (setq K (1+ K) E (ssname SA K))
               (if COMPROBLIC
                   (progn
                     A (cos (cdr (assoc 51 (entget E*))))
                     (if (not (equal A 1 Q0))
                         (progn
                           (setq LE (entget E)
                                 LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A))
                                           (assoc 40 LE) LE)
                                 LE (subst (cons 41 (* (cdr (assoc 41 LE)) A))
                                           (assoc 41 LE) LE))
                           (entmod LE)))
                     (setq E* (entnext E*))))
               (ssadd E SS)))))
```

Veiem de nou **REDEF-BLOC*S**, on hem subratllat els dos accessos a **INSPARCIAL**,

```
(defun REDEF-BLOC*S (/ LB)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (setq J (1+ (strlen BLOC-1)))
  (tblnext "BLOCK" T)
  (while (setq LB (tblnext "BLOCK"))
         (if (wcmatch (setq BLOC-1* (cdr (assoc 2 LB)))
                      (strcat BLOC-1 M (substr M 2)))
             (progn
               (setq E (cdr (assoc -2 LB)) LE (entget E)
                     K (cdr (assoc 41 LE))
                     BLOC-2* (strcat BLOC-2 (substr BLOC-1* J)))
               (command "SCP" "EZ" O (cdr (assoc 210 LE)))
               (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI) ())
               (command "SCP" "PR"
                        "BLOQUE" BLOC-1* O (setq E (ssname SS 0)) ""
                        "BLOQUE" BLOC-2* O (ssdel E SS) "")))))
```

```
  (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
  (tblnext "BLOCK" T)
  (while (setq LB (tblnext "BLOCK"))
         (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                      (strcat N "*"))
             (progn
               (setq E (cdr (assoc -2 LB)) LE (entget E)
                     BLOC-1 (cdr (assoc 2 LE))
                     LB (substr BLOC-1 J)
                     BLOC-2 (strcat BLOC-2* LB))
               (if (> LB "") (command "SCP" "EZ" O (cdr (assoc 210 LE))))
               (INSPARCIAL (cdr (assoc 41 LE)) (cdr (assoc 42 LE))
                           (cdr (assoc 43 LE)) (/ (* (cdr (assoc 50 LE)) 180) PI)
                           (> LB ""))
               (if (> LB "") (command "SCP" "PR"))
               (command "BLOQUE" BLOC* O SS ""))))
  (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2*)
  (setvar "CLAYER" CAPA))
```

i també **INSERT***, on hem fem el mateix amb els tres accessos:

```
(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** Z-XY* XY XY* XY** OXY* OXY** SA SS
                  OO-1 OO-2 BLOC-1 BLOC-2 BLOC* BLOC-1* BLOC-2* BL*EX)
  (setq BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X
            Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'OX* 'OX** 'Y**)
        (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
              A (inters Z-XY (polar Z-XY W 1) O X** ())
              Z-XY* (polar A (angle A Z-XY)
                           (/ (* (distance A Z-XY) (distance X* X**))
                              (distance X X**))))))
  (if NORM-Z
      (setq BLOC* (strcat BLOC "_"))
      (progn
        (setq X (inters X* Y* O Z-XY* ())
              X (if X X Z-XY*)
              Z-XY (- (angle O X) (angle Y* X*))
              Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
              Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
              Z (trans (list (car Z-XY*) (cadr Z-XY*) (last Z)) 1 0))
        (command "SCP" "Z" O X
                 "SCP" "X" (* I 90))
        (setq XY (list OX* 0 0) Z (trans Z 0 1))
        (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'OXY* 'OXY** 'Z**)
        (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                            (FIX+ (/ OXY** OXY* Q0)))
              BLOC* (strcat BLOC K)
              BLOC-1* (strcat BLOC-1 K) BLOC-2* (strcat BLOC-2 K)
              BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
        (command "SCP" "PR" "SCP" "PR")
        (if (not BL*EX)
            (progn
              (INSPARCIAL (setq K (/ OXY* OXY**)) K K X* ())
              (command "SCP" "Z" O X
                       "SCP" "X" 90
                       "GIRA" SS "" O XY*
                       "SCP" "Z" O XY**
                       "BLOQUE" BLOC-1* O (setq E (ssname SS 0)) ""
```

```
                        "BLOQUE" BLOC-2* 0 (ssdel E SS) ""
                        "SCP" "PR")))
        (setq B BL*EX BLOC BLOC* BLOC-1 BLOC-1* BLOC-2 BLOC-2*
              BLOC* (strcat BLOC "_" (FIX+ (/ OXY** OX* Q0))))))
  (if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
  (if (or NORM-Z BL*EX) (setq BL*EX (tblsearch "BLOCK" BLOC*)))
  (if (not BL*EX)
      (progn
        (if NORM-Z
            (INSPARCIAL (setq K (/ OX* OX**)) K K X* ())
            (progn
              (if B (command "SCP" "Z" 0 X
                             "SCP" "X" 90))
              (INSPARCIAL (setq K (/ 1 OX**) E (* OXY** K))
                          (* (distance Z Z**) K) (* E (/ OX* OXY*)) XY** T)
              (command "SCP" "PR" "SCP" "PR")))
        (command "SCP" "Z" 0 X**
                 "BLOQUE" BLOC* 0 SS ""
                 "SCP" "PR")))
  (command "CELWEIGHT" GLIN
           "CELTYPE" TLIN
           "CECOLOR" COL
           "CLAYER" CAPA)
  (eval (append '(command "INSERT" BLOC* 0 "XYZ" OX** (distance Y Y**)
                          (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1) OX** I) X**)
                (if ATREQ WWAA))))
```

Per cert: observeu com aquestes versions de **REDEF-BLOC*S** i **INSERT*** han guanyat en simplicitat en relació a les aparicions precedents, en suposar que la variable de sistema ***EXPERT*** l'hem passada a **2**; i en **INSERT***, a més, destaquem la intervenció de la funció **FIX+** en la composició dels noms **BLOC-1***, **BLOC-2*** i **BLOC*** (n'hauríem de dir **BLOC****), per arrodonir millor els valors numèrics i garantir que s'expressaran amb el mateix nombre de dígits. Però tornem al cinquè argument **COMPROBLIC**, perquè, en no haver deixat prou clar què calia entendre en 3D per "insercions obliqües", potser ho hem fet massa complicat.

A les versions 14+ i 15+ (2D) parlàvem d'insercions obliqües per referir-nos a les realitzades amb escalat no uniforme ($\mathbf{E_x \neq E_y}$), i vèiem com, en descompondre'n una en què el bloc tingués atributs editables definits amb caràcters oblics, aquests caràcters s'aixafàven. Implícitament donàvem per fet que, sempre que $\mathbf{E_x = E_y}$, un escalat diferent en la direcció ***Z*** (que en 2D mai no es produïa, perquè $\mathbf{E_z = E_x}$) hauria estat irrellevant en la geometria d'uns objectes com els atributs, que se situen en el pla ***XY*** del bloc.

En passar a 3D, aquesta hipòtesi quadrava amb el resultats empírics en la gestació de **BLOC-2*** i de **BLOC***, casos en què la inserció explosionada de **BLOC-2** s'havia realitzat amb escalat uniforme: deixant de banda $\mathbf{E_z}$, allò que interessava era que els atributs pertanyien al pla ***XY*** de **BLOC-2** i que $\mathbf{E_x = E_y}$. Però no quedava gens clar si seguia quadrant-hi quan definíem **BLOC**** sobre la inserció explosionada de **BLOC-2*** (pel que fa als atributs), realitzada amb factors d'escala $\mathbf{E_y \neq E_x \neq E_z}$. D'una banda, el resultat empíric incontestable era que, si els atributs editables s'havien definit amb caràcters oblics, l'aixafament es produïa sempre, i davant d'això podíem adoptar dues posicions: la primària, consistent a veure en això una confirmació més de la hipòtesi de treball (l'aixafament es produïa precisament perquè $\mathbf{E_x \neq E_y}$), o la de buscar tres peus al gat i argumentar que, ben mirat, la inserció no comportava canvi d'obliqüitat dels atributs sinó just tot el contrari (els atributs de **BLOC-2*** no pertanyen al seu pla ***XY*** sinó al pla perpendicular que forma un angle díedre β_1 amb el pla ***ZX***, i si havíem adoptat un factor compensatori $\mathbf{E_z \neq E_x}$ era per mantenir quadrada la base del cub associat a **BLOC**, on se situaven els atributs), raó per la qual la hipòtesis no era vàlida, ni més no tal i com la interpretàvem.

Doncs bé, en el capítol precedent ja aconsellàvem no trencar-se el cap intentant treure l'entrellat d'una resposta anòmala que, al cap i a la fi, no era més que una fallada del sistema. Deixem-nos d'explicacions lògiques i acceptem sense més que l'efecte d'aixafament es donarà quan $\mathbf{E_x \neq E_y}$ o quan $\mathbf{E_x \neq E_z}$. I, per veure que és simplement això i que no està condicionat a la deformació dels atributs, provem d'inserir un **BLOC** amb atributs P i N escrits amb caràcters oblics, amb els factors

$E_y = E_x \neq E_z$: hi haurà aixafament, cosa que prova que l'efecte no està condicionat al canvi d'obliqüitat dels caràcters, circumstància que òbviament aquí no es dóna. De forma que eliminarem l'últim argument de **INSPARCIAL**: això no vol dir prescindir de **COMPROBLIC** sinó que, de ser un paràmetre-valor (el receptacle d'aquest cinquè argument) passa a variable local, amb un valor que només dependrà de l'acompliment de **$E_x = E_y = E_z$**. Però atenció: no ens deixem endur per la rutina i cometem l'error d'escriure

```
(setq COMPROBLIC (not (and (equal EX EY Q0) (equal EX EZ Q0))))
```

perquè això no té res a veure amb el grau de precisió que vulguem introduir en el procés, mitjançant la variable **Q0**, sinó que és una condició que detecta AutoCAD amb la seva pròpia precisió de càlcul (remarquem, a més, que no n'hi ha prou amb la igualtat dels tres valors absoluts, sinó que el signe també és significatiu); així, que haurem de fer

```
(setq COMPROBLIC (not (= EX EY EZ)))
```

Si considerem els continguts comuns a **INSPARCIAL** i a la funció **C:DESCOMPOK**, que no ens molestem a reproduir per ser gairebé idèntica a la de la VERSIÓ 15+ (només cal ampliar el segon argument de **wcmatch**, que de ser **(strcat "*_" N)** passarà a ser **(strcat "*_" N ",*_" N N "_*")**, i la condició d'accés al salt que en la VERSIÓ 14+ havíem individualitzat com a funció **PASSA-ATTRIB**, que de ser **(not *EDITATRIBS-P*)** passarà a ser **(and (not *EDITATRIBS-P*) (wcmatch BLOC* (strcat "*_" N))** perquè, si decidim situar els atributs P a **BLOC-1**, "ATTRIB" i "SEQEND" únicament acompanyaran la inserció d'aquest bloc, no la de **BLOC-1***), es veu la conveniència d'agrupar-los en una funció que anomenarem **REST-ATRIBS**, per evitar reiteracions. Però millor que **REST-ATRIBS** no ho controli tot, sense mitjancers, perquè si no se'ns convertirà en un monstre farcit de sentències condicionals, en funció de les variables booleanes **COMPROBLIC** (ja comentada) i **DESCOMP** (nova, que indica si l'accés a **REST-ATRIBS** es produeix o no des de **C:DESCOMPOK**):

```
(defun REST-ATRIBS (DESCOMP)
  (setq COMPROBLIC (if DESCOMP T COMPROBLIC) K -1)
  (if COMPROBLIC
      (progn
        (setq E* (if DESCOMP BLOC* BLOC-2)
              E* (cdr (assoc -2 (tblsearch "BLOCK" E*))))
        (if (and DESCOMP (not *EDITATRIBS-P*) (wcmatch BLOC* (strcat "*_" N)))
            (while (wcmatch (cdr (assoc 0 (entget (entnext E*))))
                            "ATTRIB,SEQEND")
                   (setq E* (entnext E*))))))
  (while (setq K (1+ K)
               E (ssname SA K)
               E* (if COMPROBLIC
                      (if DESCOMP (entnext E*) E*)
                      E))
         (if COMPROBLIC
             (progn
               (setq A (cos (cdr (assoc 51 (entget E*))))
                     E* (if DESCOMP E* (entnext E*)))
               (if (not (equal A 1 Q0))
                   (progn
                     (setq LE (entget E)
                           LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A))
                                     (assoc 40 LE) LE)
                           LE (subst (cons 41 (* (cdr (assoc 41 LE)) A))
                                     (assoc 41 LE) LE))
                     (entmod LE)))))
         (if (not DESCOMP) (ssadd E SS))))
```

Això sí: tant **INSPARCIAL**

```
(defun INSPARCIAL (EX EY EZ ANG / COMPROBLIC)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast))
        COMPROBLIC (not (= EX EY EZ)))
  (if OO-2
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
```

```
                "ATTREQ" (if ATREQ 1 0)
                "DESCOMP" (entlast))
        (setq SA (ssget "P"))
        (REST-ATRIBS ())))))
```

com **C:DESCOMPOK** (on haurem realitzat petits retocs uniformitzadors, substituint la variable **SS** per **SA** i **N** -en funcions de comptador- per **K**)

```
(defun C:DESCOMPOK (/ Q0 ECO BLOC* E E* LE SA A K N)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D, sin deformación de ")
  (prompt "sus atributos")
  (prompt "\n(si se definieron con caracteres oblicuos, usando DESCOMP ")
  (prompt "reducirían su altura):")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              ECO (getvar "CMDECHO"))
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (setvar "CMDECHO" ECO)
        (if (and (wcmatch BLOC* (strcat "*_" N ",*_" N N))
                 (setq SA (ssget "P")))
            (REST-ATRIBS T)))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

hauran quedat considerablement reduïdes. Tanmateix, amb una longitud de codi molt semblant, serà més intel·ligible reequilibrar la càrrega entre la funció servidora (que en consonància amb la variable booleana **COMPROBLIC**, ara anomenarem **REST-OBLIC** en comptes de **REST-ATRIBS**) i les usuàries **INSPARCIAL** i **C:DESCOMPOK**, tal i com us ho trobareu en la versió completa del codi.

Us la oferiríem ja sense més preàmbuls, si no fos perquè encara hem de complir una promesa implícita pàgines enrera, quan anunciàvem que la VERSIÓ 16+ i successives integrarien un aspecte formal que fins ara havíem descuidat o que, més exactament, sols havíem plantejat en OPTIMITZACIÓ DELS RECURSOS EXISTENTS: 2 EXEMPLES, capítol preliminar d'aquesta segona part: l'abast de l'execució de les ordres ***DESHACER*** i ***H*** després de ***INS2D***/***INS3D***. El fet que el pas de 2D a 3D (de la VERSIÓ 15+ a la VERSIÓ 16+) impliqués la renúncia a l'elegant i econòmica solució de deixar en mans de l'Editor de Dibuix l'assignació de valor als atributs N, just en acabar ***INSERTOK*** amb una ordre ***INSERT*** a mig executar (decisió justificada per imperatiu d'un canvi de ***SCP*** que resultava imprescindible per organitzar la concatenació d'insercions), ens ha donat l'oportunitat de fer nostre allò de *no hay mal que por bien no venga*, perquè gràcies a haver hagut d'acceptar la simulació **VAL-ATRIBS** i les llistes **WWAA** com un mal necessari, completant així els arguments de l'última expressió **(command "INSERT" ...)** avaluada, ara podem posar sobre la taula i abordar aquest tema: una qüestió merament formal, si es vol, però de força repercussió funcional. En aquell capítol preliminar, sobretot en relació amb **C:GINSERT**, es presentava la disjuntiva entre fer primer **(setvar "CMDECHO" 0)** i després **(command "DESHACER" "I")** (cas en què, si abans d'usar l'aplicació teníem ***CMDECHO*** = **1**, en executar després l'ordre ***H*** ens quedàvem amb ***CMDECHO*** = **0**) o bé fer primer **(command "DESHACER" "I")** i després **(setvar "CMDECHO" 0)** (en aquest cas, ***H*** restaurava ***CMDECHO*** = **1** però ***GINSERT*** deixava el rastre de l'execució, *DESHACER Indique el número de operaciones a deshacer o [Auto/Control /Inicio/Fin/Marca/Retorno]<1>: I*, a més d'un canvi de línia seguit de *Comando:*). La intervenció d'un nombre indeterminat d'arguments **PAUSE** encara ho complicava més, però a ***INS2D***/***INS3D*** el problema de base subsisteix i tenim dues opcions, que aplicarem a la segona estratègia:

- Recórrer a Visual LISP per instal·lar un reactiu de tipus **vlr-COMMAND-reactor**, sensible als successos **:vlr-CommandWillStart** i **:vlr-CommandEnded**, associats a les funcions de resposta **–ECO-1** i **–ECO-2** i que, gestionant la variable **CTRL–ECO** (global en tota la sessió de dibuix) en combinació amb la funció **SENSE-RASTRE**, sobreescriguin espais en blanc sobre l'eco de ***DESHACER*** i el de l'opció ***I***, i sobre *Comando:*. Amb **:vlr-CommandEnded** i **–ECO-2** n'hi hauria prou, però, posats a manllevar aquest dispositiu del capítol esmentat, més val endur-se'l sencer per

donar cobertura a les aplicacions ***GINSERT*** i ***RATREDIT*** que allà havíem preparat (de cara a la segona caldria afegir un reactiu tipus **vlr-ACDB-reactor**, sensible al succés **:vlr-ObjectOpenedForModify**, associat a la funció de resposta **SEL-EDIT**) i que així podríem incloure en l'arxiu .LSP on guardem ***INS2D***, ***INS3D*** i ***DESCOMPOK***.
- Desplegar uns recursos més modestos però que condueixen a resultats semblants: sense necessitat de reactius, la funció **QUASI-SENSE-RASTRE** emmascara l'eco de ***DESHACER*** i el missatge *Comando:;* només quedarà dempeus l'irreductible eco de ***I***.

En el codi que veurem tot seguit s'ha adoptat la segona opció (però deixant la primera a l'abast de l'usuari, incorporada com a comentari), perquè la *I* passarà gairebé desapercebuda a continuació de l'últim ínput introduït: tant de bo els missatges *Verificar valores de atributos* fossin tan discrets!

Com que amb un simple desplaçament de l'accés a la funció **VAL-ATRIBS** (respecte al lloc que ocupava en la VERSIÓ 15+) hem aconseguit que l'assignació de valor als atributs es produeixi abans no s'activi el dispositiu ***DESHACER Inicio*/*Fin***, i això garanteix que cap incidència de mecanografiat no repercutirà negativament sobre l'eficàcia de l'esmentat dispositiu, no caldrà que ens compliquem la vida cercant la manera d'escapar del foc sense caure a les brases, que així de bizantina era a ***RATREDIT*** la discussió sobre l'alternativa d'ús **getstring**/**GETSTR**: **getstring** (que és l'instrument de **VATR**, al seu torn usada per **VAL-ATRIBS**) no causarà cap problema.

Per raons que se'ns escapen, la presència d'un genèric **BLOC-2*** de contingut nul, que correspon a un **BLOC** sense atributs N (ni P, si ***EDITATRIBS-P*** és **T**), dins de l'expressió

```
(command "BLOQUE" BLOC-2* O (ssdel E SS) "")
```

present a **REDEF-BLOC*S** i a **INSERT***, impedeix que una ordre ***H*** executada després de ***INS2D*/*INS3D*** anul·li de cop totes les operacions que aquesta inclou. No sembla que la causa sigui un error provocat per l'expressió **(ssdel E SS)**, perquè complicant-la una mica, per evitar-lo, segueix produint-se el fenòmen:

```
(eval (append '(command "BLOQUE" BLOC-2* O) (if (> (sslength SS) 1)
                                                '((ssdel E SS)))
                                            '("")))
```

Per obviar-lo, la funció **PRO-DESHACER-I/F**, que de passada integrarà la línia

```
(command "BLOQUE" BLOC-1* O (setq E (ssname SS 0)) "")
```

que la precedeix en totes dues aparicions, diversificarà els procediments: quan hi hagi atributs s'usarà l'esmentada expressió, i quan no n'hi hagi recorrerem a la funció **MAKEBLOC** (**entmake**); si no seguim sempre la segona via és per no haver de visualitzar missatges *Redefiniendo el bloque "ATRIBS_DE_..."* més que quan sigui estrictament necessari (quan ***CMDECHO*** = **0**, les redefinicions executades via **command** són silencioses, però si treballem amb **entmake** no).

Ara sí, sensa més dilatòries, vinga el codi complet!:

```
; VERSIÓ 16+

(setq *EDITATRIBS-P* T) ; Si voleu que els atributs predefinits siguin editables.

;;; (vl-load-com)
;;; (vlr-remove-all)
;;; (vlr-COMMAND-reactor ()'((:vlr-CommandWillStart . -ECO-1)
;;;                          (:vlr-CommandEnded . -ECO-2)))

;;; (defun -ECO-1 (N-REACTIU L-ORDRE)
;;;   (if CTRL--ECO (BS (1+ (strlen (getcname (strcat "_" (car L-ORDRE))))))))

;;; (defun -ECO-2 (N-REACTIU L-ORDRE) (if CTRL--ECO (BS (1+ (strlen CTRL--ECO)))))

;;; (defun SENSE-RASTRE ()
;;;   (command "DESHACER" (progn (if (= ECO 1)
;;;                                  (progn
;;;                                    (BS 100)
;;;                                    (setq CTRL--ECO "I")))
;;;                              "I"))
;;;   (if (= ECO 1)
;;;       (progn
;;;         (BLANC)
;;;         (setq CTRL--ECO ()))))
```

```
(defun QUASI-SENSE-RASTRE ()
  (command "DESHACER" (progn (if (= ECO 1) (BS 100)) "I"))
  (if (= ECO 1) (BLANC)))

(defun C:INS2D () (INSERTOK T))

(defun C:INS3D () (INSERTOK ()))

(defun BS (I) (repeat I (princ "\10 \10")))

(defun BLANC () (princ "\r") (repeat 100 (princ " ")) (princ "\r") (princ))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun U*V (U V)
  (list (- (* (cadr U) (last V)) (* (last U) (cadr V)))
        (- (* (last U) (car  V)) (* (car  U) (last V)))
        (- (* (car  U) (cadr V)) (* (cadr U) (car  V)))))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS (/ ICVP N M VD V LLAA)
  (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
  (while E
         (if (and (= (cdr (assoc 0 (setq LE (entget E)))) "ATTDEF")
                  (= (logand (setq ICVP (cdr (assoc 70 LE))) 10) 0))
             (progn
               (if (and (not LLAA) ATREQ)(prompt "\nIndique valores de atributo"))
               (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LE))
                     (cdr (assoc 3 LE)) (cdr (assoc 1 LE)))
               (setq LLAA (cons (list W N M V) LLAA))))
         (setq E (entnext E)))
  (setq W ())
  (foreach LA LLAA
    (if (car LA)
        (progn
          (if (and (not W) ATREQ) (prompt "\nVerificar valores de atributo"))
          (VATR (cons "" W) (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq WWAA (cons V WWAA)))
  (setq WWAA (append WWAA W)))
```

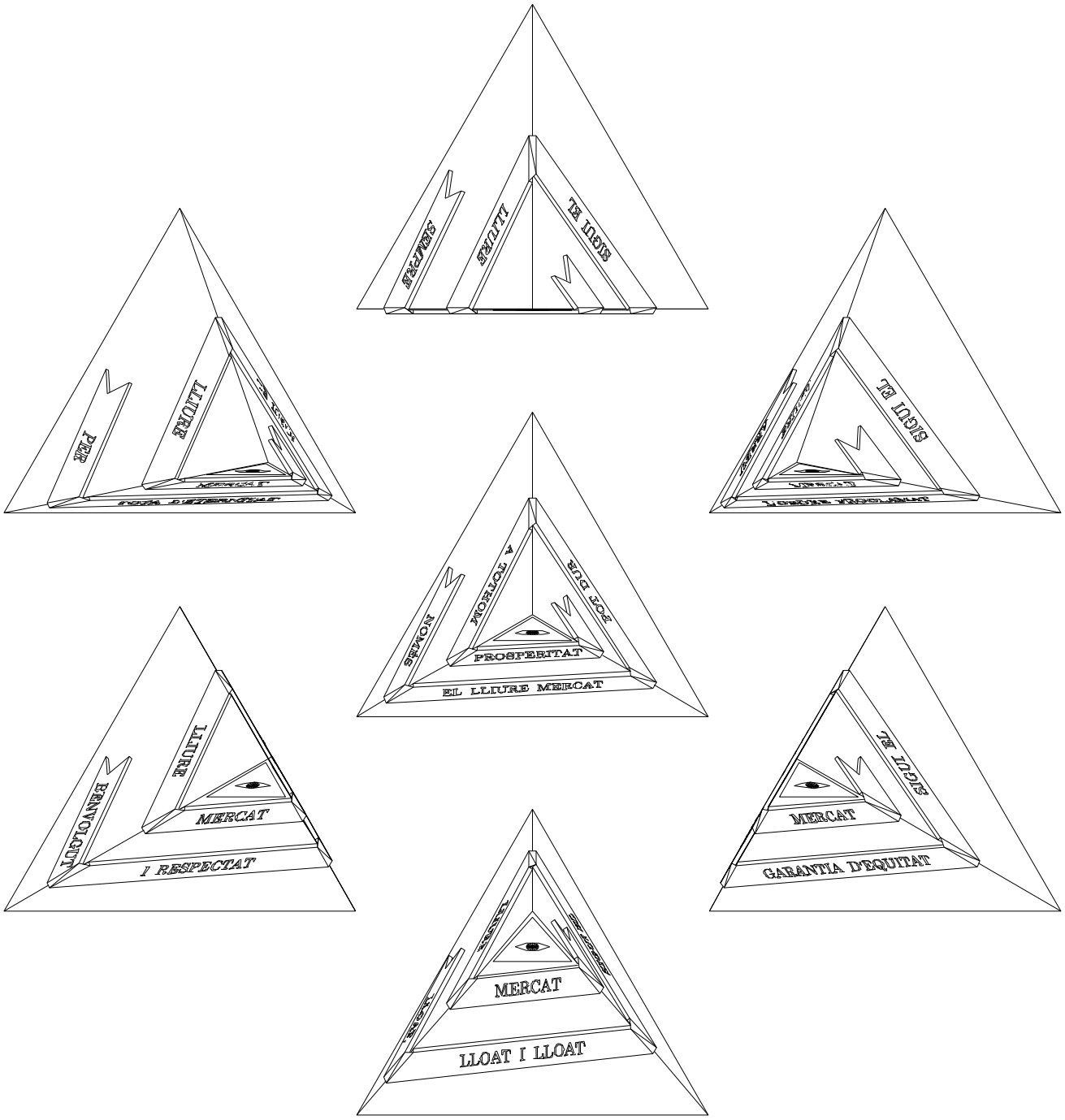
```
(defun MAKEBLOC (BLOC/2 OO ATRIBS / C-10 C-11)
  (entmake (list '(0 . "BLOCK") (cons 2 BLOC/2) '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun LINIA-BASE ()
  (polar (cdr (assoc 11 LE))
         (- (cdr (assoc 50 LE)) PI/2)
         (* (cdr (assoc 40 LE))
            (if (= C-74 1)
                (/ 1.0 -3)
                (if (= C-74 2) 0.5 1)))))

 (defun REST-OBLIC ()
  (setq A (cos (cdr (assoc 51 (entget E*)))))
  (if (not (equal A 1 Q0))
      (progn
        (setq LE (entget E)
              LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A)) (assoc 40 LE) LE)
              LE (subst (cons 41 (* (cdr (assoc 41 LE)) A)) (assoc 41 LE) LE))
        (entmod LE))))

(defun INSPARCIAL (EX EY EZ ANG / COMPROBLIC)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast))
        COMPROBLIC (not (= EX EY EZ)))
  (if OO-2
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast))
        (setq SA (ssget "P") K -1
              E* (if COMPROBLIC (cdr (assoc -2 (tblsearch "BLOCK" BLOC-2)))))
        (while (setq K (1+ K) E (ssname SA K))
               (if COMPROBLIC
                   (progn
                     (REST-OBLIC)
                     (setq E* (entnext E*))))
               (ssadd E SS)))))

(defun PRO-DESHACER-I/F ()
  (command "BLOQUE" BLOC-1* O (setq E (ssname SS 0)) "")
  (if OO-2
      (command "BLOQUE" BLOC-2* O (ssdel E SS) "")
      (MAKEBLOC BLOC-2* () ())))

(defun REDEF-BLOC*S (/ LB)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (setq J (1+ (strlen BLOC-1)))
  (tblnext "BLOCK" T)
  (while (setq LB (tblnext "BLOCK"))
         (if (wcmatch (setq BLOC-1* (cdr (assoc 2 LB)))
                      (strcat BLOC-1 M (substr M 2)))
             (progn
               (setq E (cdr (assoc -2 LB)) LE (entget E)
                     K (cdr (assoc 41 LE))
                     BLOC-2* (strcat BLOC-2 (substr BLOC-1* J)))
               (command "SCP" "EZ" O (cdr (assoc 210 LE)))
               (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
               (command "SCP" "PR")
               (PRO-DESHACER-I/F))))
  (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
```

```
  (tblnext "BLOCK" T)
  (while (setq LB (tblnext "BLOCK"))
         (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                      (strcat N "*"))
             (progn
               (setq E (cdr (assoc -2 LB)) LE (entget E)
                     BLOC-1 (cdr (assoc 2 LE))
                     LB (substr BLOC-1 J)
                     BLOC-2 (strcat BLOC-2* LB))
               (if (> LB "") (command "SCP" "EZ" O (cdr (assoc 210 LE))))
               (INSPARCIAL (cdr (assoc 41 LE))
                           (cdr (assoc 42 LE))
                           (cdr (assoc 43 LE))
                           (/ (* (cdr (assoc 50 LE)) 180) PI))
               (if (> LB "") (command "SCP" "PR"))
               (command "BLOQUE" BLOC* O SS ""))))
  (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2*)
  (setvar "CLAYER" CAPA))

(defun SEGR-ATRIBS (/ BL1EX BL2EX AT AT-CP C-72 C-74)
  (setq BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2))
  (if (and BL1EX BL2EX)
      (setq OO-2 (/= (cdr (assoc 0 (entget (cdr (assoc -2 BL2EX))))) "ENDBLK"))
      (progn
        (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (cdr (assoc 70 LE))
                                   (if *EDITATRIBS-P* 2 10)) 0))
                   (setq C-72 (cdr (assoc 72 LE)) C-74 (cdr (assoc 74 LE))
                         LE (if (= C-72 4)
                                (subst (cons 72 (setq C-72 1))
                                       (cons 72 4)
                                       (subst (cons 74 (setq C-74 2))
                                              (assoc 74 LE)
                                              LE))
                                LE)
                         LE (if (and (< C-72 3) (> C-74 0))
                                (subst (cons 74 0)
                                       (assoc 74 LE)
                                       (subst (cons 11 (LINIA-BASE))
                                              (assoc 11 LE)
                                              LE))
;;                                       (subst (cons 11 (LINIA-BASE))
;;                                              (assoc 11 LE)
;;                                              LE))
;;   Podeu usar les 3 línies precedents com alternativa a les 5 línies subsegüents
                                       (if (> C-72 0)                          ;;
                                           (subst (cons 11 (LINIA-BASE))       ;;
                                                  (assoc 11 LE)                ;;
                                                  LE)                          ;;
                                           LE))                                ;;
                                LE)
                         OO-2 (cons LE OO-2))
                   (setq OO-1 (cons LE OO-1)
                         AT-CP (if AT-CP T AT)))
               (setq E (entnext E)))
        (command "REGENT"
                 "CECOLOR" "PORCAPA"
                 "CELTYPE" "PORCAPA"
                 "CELWEIGHT" -1)
        (MAKEBLOC BLOC-1 OO-1 AT-CP)
        (MAKEBLOC BLOC-2 OO-2 OO-2)
        (if BL1EX (REDEF-BLOC*S))
        (setvar "CECOLOR" COL)
        (setvar "CELTYPE" TLIN)
        (setvar "CELWEIGHT" GLIN))))
```

```
(defun CALCULA (U V OV AGUT U* U** OU* OU** V**)
  (setq A (polar O (- (angle O V) PI/2) OV)
        W (if AGUT (angle A U) (angle U A))
        B (mapcar '/ (mapcar '+ A U) '(2 2 2))
        W (angle O (polar B W (distance O B))))
  (set U* (inters U (polar U W 1) O B ()))
  (set U** (inters O (polar O (+ W PI/2) 1) U (eval U*) ()))
  (set OU* (distance O (eval U*)))
  (set OU** (distance O (eval U**)))
  (set V** (inters V (polar V W 1) O (eval U**) ())))

 (defun FIX+ (O / N)
  (setq N (itoa (if (> (- O (setq N (fix O))) 0.5) (1+ N) N)))
  (repeat (- (strlen M) 1 (strlen N)) (setq N (strcat "0" N)))
  N)

(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** Z-XY* XY XY* XY** OXY* OXY** SA SS
                  OO-1 OO-2 BLOC-1 BLOC-2 BLOC* BLOC-1* BLOC-2* BL*EX)
  (setq BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X
            Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'OX* 'OX** 'Y**)
        (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
              A (inters Z-XY (polar Z-XY W 1) O X** ())
              Z-XY* (polar A (angle A Z-XY)
                           (/ (* (distance A Z-XY) (distance X* X**))
                              (distance X X**))))))
  (if NORM-Z
      (setq BLOC* (strcat BLOC "_"))
      (progn
        (setq X (inters X* Y* O Z-XY* ()) X (if X X Z-XY*)
              Z-XY (- (angle O X) (angle Y* X*))
              Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
              Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
              Z (trans (list (car Z-XY*) (cadr Z-XY*) (last Z)) 1 0))
        (command "SCP" "Z" O X
                 "SCP" "X" (* I 90))
        (setq XY (list OX* 0 0) Z (trans Z 0 1))
        (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'OXY* 'OXY** 'Z**)
        (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                            (FIX+ (/ OXY** OXY* Q0)))
              BLOC* (strcat BLOC K)
              BLOC-1* (strcat BLOC-1 K) BLOC-2* (strcat BLOC-2 K)
              BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
        (command "SCP" "PR" "SCP" "PR")
        (if (not BL*EX)
            (progn
              (INSPARCIAL (setq K (/ OXY* OXY**)) K K X*)
              (command "SCP" "Z" O X
                       "SCP" "X" 90
                       "GIRA" SS "" O XY*
                       "SCP" "Z" O XY**)
              (PRO-DESHACER-I/F)
              (command "SCP" "PR")))
        (setq B BL*EX BLOC BLOC* BLOC-1 BLOC-1* BLOC-2 BLOC-2*
              BLOC* (strcat BLOC "_" (FIX+ (/ OXY** OX* Q0))))))
  (if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
  (if (or NORM-Z BL*EX) (setq BL*EX (tblsearch "BLOCK" BLOC*)))
```

```
  (if (not BL*EX)
      (progn
        (if NORM-Z
            (INSPARCIAL (setq K (/ OX* OX**)) K K X*)
            (progn
              (if B (command "SCP" "Z" O X
                             "SCP" "X" 90))
              (INSPARCIAL (setq K (/ 1 OX**) E (* OXY** K))
                          (* (distance Z Z**) K) (* E (/ OX* OXY*)) XY**)
              (command "SCP" "PR" "SCP" "PR")))
        (command "SCP" "Z" O X**
                 "BLOQUE" BLOC* O SS ""
                 "SCP" "PR")))
  (command "CELWEIGHT" GLIN
           "CELTYPE" TLIN
           "CECOLOR" COL
           "CLAYER" CAPA)
  (eval (append '(command "INSERT" BLOC* O "XYZ" OX** (distance Y Y**)
                          (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1) OX** I) X**)
                (if ATREQ WWAA))))

(defun INSERTOK (2D / Q0 2*PI PI/2 M N CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ
                      EXPERT BL BLOC WWAA O X Y Z UV Z-XY OX OY OZ X-Y X-O-Y A B
                      E LE E* I J K W NORM-XY NORM-Z)
  (setq Q0 0.001 M "_")
  (repeat (strlen (itoa (fix (/ 0.1 Q0))))
          (setq M (strcat M "#")))
  (setq 2*PI (* PI 2) PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        EXPERT (getvar "EXPERT")
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        N (strcat BLOC M) W T)
  (while W
    (setq O (getpoint "\nPrecise punto de inserción: "))
    (foreach P (if 2D '("X" "Y") '("X" "Y" "Z"))
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                  " desde el punto de inserción: "))
            W (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC  " <1>: "))
            W (if W W 1))
      (set (read P) (mapcar '(lambda (CO CA) (+ CO (/ (- CA CO) W))) O A))
      (set (read (strcat "O" P)) (/ (distance O A) W)))
    (setq OZ (if 2D OX OZ)
          W (if (equal (setq UV (U*V (mapcar '- X O)(mapcar '- Y O))) '(0 0 0) Q0)
                " e Y están alineados."
                (if (and (not 2D) (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                                (mapcar '* UV (mapcar '- Z O)))
                                         0 Q0))
                    ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq Z (trans (if 2D (mapcar '+ O UV) Z) 1 0)
        Z-XY Y Y (trans Y 1 0))
  (if (= (logand (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) 2) (VAL-ATRIBS))
```

```
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
  (setvar "CMDECHO" 0)
  (command "OSMODE" 0
           "ATTDIA" 0
           "EXPERT" 2
           "SCP" "3" O X Z-XY)
  (setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1)
        Z-XY (list (car Z) (cadr Z) 0) I (if (< (last Z) 0) -1 1)
        NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                       (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
        NORM-Z (equal Z-XY O Q0)
        BL BLOC)
  (if (and NORM-XY NORM-Z)
      (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                    (if ATREQ WWAA)))
      (INSERT*))
  (command "SCP" "PR"
           "INSNAME" BL
           "EXPERT" EXPERT
           "ATTDIA" ATDIA
           "OSMODE" OSN
           "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))

(defun C:DESCOMPOK (/ Q0 ECO BLOC* E LE E* SA A K N)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D, sin deformación de ")
  (prompt "sus atributos")
  (prompt "\n(si se definieron con caracteres oblicuos, usando DESCOMP ")
  (prompt "reducirían su altura):")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              ECO (getvar "CMDECHO"))
        (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
      ;;; (SENSE RASTRE)     ; i activa les que comencin amb ";;;".
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (setvar "CMDECHO" ECO)
        (if (and (wcmatch BLOC* (strcat "*_" N ",*_" N N))
                 (setq SA (ssget "P")))
            (progn
              (setq E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))) K -1)
              (if (and (not *EDITATRIBS-P*) (wcmatch BLOC* (strcat "*_" N)))
                  (while (wcmatch (cdr (assoc 0 (entget (entnext E*))))
                                  "ATTRIB,SEQEND")
                    (setq E* (entnext E*))))
              (while (setq K (1+ K)
                           E (ssname SA K)
                           E* (entnext E*))
                (REST-OBLIC))))
        (command "DESHACER" "F")
        (setvar "CMDECHO" ECO))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```



SEMPRE GLORIFICAT
SIGUI EL LLIURE MERCAT

PER TOTA L'ETERNITAT
SIGUI EL LLIURE MERCAT

ARREU L'ORDRE PROCLAMAT
SIGUI EL LLIURE MERCAT

NOMÉS EL LLIURE MERCAT
POT DUR A TOTHOM PROSPERITAT

BENVOLGUT I RESPECTAT
SIGUI EL LLIURE MERCAT

ÚNICA GARANTIA D'EQUITAT
SIGUI EL LLIURE MERCAT

LLOAT, LLOAT I LLOAT
SIGUI EL LLIURE MERCAT

ENCAIX 3D DE BLOCS AMB ATRIBUTS SITUATS LLIUREMENT

Quan pensem en blocs definits i inseribles en 3D ho fem en el doble sentit de la seva composició (objectes 3D i també objectes 2D no necessàriament situats en el pla coordenat ***XY*** del sistema de referència del bloc, sinó fora d'ell, mantenint-s'hi paral·lels o amb qualsevol orientació) i de la possibilitat d'inserir-les amb escales $\mathbf{E_z \neq E_x}$. Tanmateix, quan hi intervenen atributs no Constants (Predefinits i Normals) la llibertat de posició i orientació d'aquests objectes 2D no és total. És clar que podeu situar-los on vulgueu, però les insercions no quedaran gens bé. En el cas d'insercions simples (***INSERT***) n'hi ha prou que els atributs P i N siguin paral·lels al pla ***XY***; dels que no ho siguin, només el punt base de l'atribut se situarà correctament i, pel que fa a la resta de característiques geomètriques, l'atribut s'orientarà, en relació al ***SCP*** vigent en inserir el bloc (l'actual), no com ho estava en relació al ***SCP*** vigent en definir aquest bloc sinó en relació al vigent en definir l'atribut. Però si tenim en compte que, quan explosionem una inserció en què no sigui $\mathbf{E_x = E_y = E_z}$, només els atributs situats en plans que passin pel punt base del bloc quedaran reubicats correctament (perdent el valor assignat, es clar), perquè els demés tindran bé l'escalat i l'orientació però experimentaran un desplaçament en direcció ***Z*** igual al valor $\mathbf{E_z}$, no ha d'estranyar que a les insercions múltiples obtingudes amb ***INS3D*** encara hi hagi més enrenou. Efectivament, en tractar-se d'una concatenació d'insercions del tàndem substitut de **BLOC**, **BLOC-1** i **BLOC-2**, en què de primer inserirem aquests blocs complementaris creant **BLOC-1*** amb la inserció de **BLOC-1** i **BLOC-2*** amb el producte de la inserció explosionada de **BLOC-2**, i després inserim aquests genèrics de primer grau creant **BLOC**** amb la inserció de **BLOC-1*** i la inserció explosionada de **BLOC-2*** plegades, per força tots els atributs P i N en resultaran afectats, tret dels situats en el mateix pla ***XY*** del bloc: els ubicats fora d'aquest pla malgrat ser-ne paral·lels, resistiran la primera inserció però no el seu explosionament; els no paral·lels ni això, perquè ja quedaran malmesos en la primera inserció.

Els atributs d'AutoCAD tenen nombroses limitacions, excepcions o fallades (que cadascú les qualifiqui com li sembli més adient) a l'hora de redefinir el bloc portador o a l'hora de editar-los, però que aquestes fins i tot afectin el "nucli dur" del sistema, és a dir, les ordres ***BLOQUE*** i ***INSERT***, ja sembla inadmissible: si ***INSERT*** no ha de poder ubicar correctament determinats atributs, ¿per què ***BLOQUE*** els admet com a components? I aquí és on, posats a pretendre superar amb ***INS2D*** i ***INS3D*** algunes de les limitacions de ***INSERT***, a costa, això sí, d'implicar-hi blocs addicionals (**BLOC-1**, **BLOC-2**, **BLOC-1***, **BLOC-2***, i **BLOC*** o **BLOC**** segons el cas), podríem imposar-nos la conquesta d'una altra fita, més enllà de les ja assolides: la inserció d'un bloc 3D amb atributs, de manera que el cub unitari de referència encaixés en un paral·lelepípede determinat (fins aquí, ja ho tenim) i que tots els components s'ajustessin mètricament a la transformació afí implícita en l'encaix, incloent-hi tots els atributs.

La solució pot implicar uns petits canvis respecte la VERSIÓ 16+ o complicar-la considerablement: tot depèn de si volem mantenir a qualsevol preu l'editabilitat dels atributs N o P, tenint en compte que amb ***-ATREDIT Sí*** només podem actuar sobre els paral·lels al pla ***XY*** del ***SCP*** actual, raó per la qual per editar un per un atributs amb orientacions diverses caldrà executar ***SCP*** i ***-ATREDIT*** reiteradament.

Si renunciem a editar els atributs, o almenys els no situats en el pla ***XY*** de **BLOC**, només caldria que **SEGR-ATRIBS** donés gat per llebre, transformant-los en atributs Constants o en textos: així, aquests components estarien representats a **BLOC-1** i no a **BLOC-2** (on s'haurien d'haver quedat els atributs P, si ***EDITATRIBS-P*** fos **T**). Les modificacions en el codi serien mínimes (sobretot en la primera opció), però no perdrem el temps presentant-les perquè no seria honest qualificar aquest nyap de solució. Ja no seria només que renunciéssim a l'editabilitat d'alguns atributs sinó que estaríem contravenint els pressupostos de partença, perquè una de dues: o bé volíem que el seu valor fos el mateix a totes les insercions, cas en què tocava haver-los incorporat directament a **BLOC** com atributs C o com a textos (segons que interessés o no fer-los aparèixer en algun llistat via ***ATREXT***); o bé volíem fer assignacions diferents, i en aquest cas la transformació implicaria que el valor assignat la primera vegada que **INSERTOK** se les veiés amb **BLOC** (fos predefinit o subministrat per l'usuari en temps d'execució, segons com ens ho haguéssim muntat) es repetiria en les successives, tret que abans de cada inserció redefiníssim **BLOC**

tot canviant aquest valor i forcéssim l'actualització de **BLOC-1** i **BLOC-2** eliminant el segon amb ***LIMPIA***. No s'hauria acabat aquí la cosa perquè, tal i com ho tenim a la VERSIÓ 16+, hauríem de prescindir de **REDEF-BLOC*S**; si no, el nou valor constant es propagaria a tots els genèrics existents i totes les insercions precedents en quedarien afectades, uniformitzant-se al nou valor.

Comprovada la vigència d'aquell aforisme que diu que tot allò que és barat acaba resultant car, descartem adaptacions minimalistes, afrontem el repte amb totes les conseqüències i respectem les característiques originals dels atributs. Tot i que, abans d'endinsar-nos per aquest camí, hem de tenir clar que caldrà seguir ampliant el seguici de **BLOC:** si més no, amb un bloc addicional per cada atribut "atípic" (des d'ara qualificarem així els atributs N o P no situats en el pla ***XY*** del bloc). Perquè si algú havia pensat a reubicar els atributs desplaçats després de cada explosionament, abans de crear el bloc genèric de grau superior, i editar els de la inserció final per corregir-ne l'orientació, que se n'oblidi: ***-ATREDIT*** només admet l'edició individualitzada dels atributs paral·lels al pla ***XY*** del ***SCP*** actual, com recordàvem fa poc, i les opcions ***Posición*** i ***ángUlo*** només actuen en el pla de cada atribut. I tampoc no servirà de res tractar d'eludir les irregularitats del procés registrant en la taula de símbols "*APPID*" els noms dels atributs atípics i definint un substitut **BLOC-2** desproveït d'ells però amb la informació que els descriu emmagatzemada, amb codi -3, com a dades ampliades (*Extended Entity Data*, canviant els codis habituals per altres del rang 1010 ... 1042) i integrada, per exemple, en l'últim dels atributs típics o, si no n'hi ha, en l'objecte virtual "*ENDBLK*": això podria funcionar en objectes convencionals, que en ser manipulats per ordres d'edició transmetrien els canvis geomètrics als paquets d'informació *EED* associats, informació actualitzada que finalment serviria per restituir els transformats dels objectes mantinguts discretament a l'ombra durant el procés; però quan parlem d'una inserció de bloc ens trobem a la base de dades del dibuix amb un trist objecte "*INSERT*", amb especificació del punt d'inserció, factors d'escala, angle de gir, orientació de l'eix ***Z*** del ***SCP*** vigent en fer la inserció (codis 10, 41, 42, 43, 50 i 210) i pareu de comptar, pel que fa a la geometria. D'altra banda, si rastregem en els components del bloc a partir del primer, fent **(entget (entnext (entnext ...(cdr (assoc -2 (tblsearch "BLOCK" BLOC-2)))...))...)**, on el lloc dels últims punts suspensius estaria ocupat per una llista amb els noms registrats a la taula "*APPID*", trobarem la mateixa informació emmagatzemada, sense cap afectació geomètrica: la que correspon als atributs atípics absents de **BLOC-2** però amb la mètrica que li correpondria en el moment de crear aquest bloc (igual a la de **BLOC**), no pas en acabar **INSERTOK**. I tampoc en treuríem res explosionant **BLOC****, ni que fos per copiar aquesta informació i restaurar la inserció fent **H:** la informació del paquet *EED* seguiria sent l'original. No hi hauria més remei que calcular en cada inserció les noves posicions, partint dels referits codis 10, 41, 42, 43, 50 i 210 o de les matrius de transformació definides pel tercer element de la llista resultant de l'expressió **(nentselp P)**, on el punt **P** hauria d'assegurar la designació gràfica de les insercions de **BLOC-1** o **BLOC-2**, de **BLOC-1*** o **BLOC-2*** i de **BLOC****: en el cas d'optar pel segon procediment podriem ometre les geometries intermèdies i anar directament de la inicial a la final, aplicant la transformació producte de les successives transformacions-insercions. Però per acabar fent això és més conseqüent mantenir el modus operandi establert, aplicant l'estratègia de l'encaix d'un quadrat **1 × 1** en un cert paral·lelogram, aprofitant els recursos disponibles (en particular, la funció **INSERT***) amb les adaptacions que calgui i, això sí inevitablement, complicant la constel·lació d'arxius auxiliars de **BLOC.** Just per no complicar encara més la troca de forma gratuïta, prescindirem de la variable ***EDITATRIBS-P*** i guardarem definitivament els atributs Predefinits en **BLOC-2**, que potser és el que hauríem d'haver fet des del moment que vam decidir repartir el contingut de **BLOC** entre **BLOC-1** i **BLOC-2** (VERSIÓ 8).

La descripció que s'ofereix tot seguit es refereix a les insercions ***INS2D***/***INS3D*** obliqües; més endavant veurem com les ortogonals, que fins ara despatxàvem amb un ***INSERT*** aplicat a **BLOC**, quan aquest tingui atributs atípics també necessitaran recórrer a arxius substituts, on la informació es classificarà d'una altra manera.

Mantindrem el **BLOC-2** amb la informació relativa als atributs N i P, peró farem que només els típics hi siguin representats per la llista d'associació que els descriu com a tals atributs a la base de dades del dibuix. Cadascun dels atípics hi estarà representat pels 3 punts (3 llistes d'associació descriptives d'objectes **"POINT"**) que corresponen a la projecció sobre el pla de l'atribut de les posicions **0,0,0**, **1,0,0** i **0,1,0** del sistema de referència de **BLOC**, és a dir, l'origen i els vectors

unitaris ***+X*** i ***+Y*** (aclarirem que el sistema de referència d'un bloc és el paral·lel al ***SCP*** vigent en el moment de la seva creació i que l'origen és el seu punt base per a insercions). Paral·lelament, crearem blocs 2D sobre el ***SCP*** definit per cada tríada de punts de **BLOC-2**, amb l'atribut com a únic contingut, i els adjudicarem un nom afegint al de **BLOC** els prefixos **"ATRIBATIP_1_DE_"**, **"ATRIBATIP_2_DE_"**, etc. Alternativament, podríem agrupar els atributs atípics situats en un mateix pla, de manera que cada un d'aquests blocs no es limités a donar suport a un únic atribut sinó a tot el grup coplanari, però no està gens clar que la complicació del codi quedés justificada per l'estalvi de memòria, atesa l'escasa freqüència amb què es presentarien aquests casos.

Això pel que fa als blocs auxiliars primaris (funció **SEGR-ATRIBS**) perquè, en la primera inserció intermèdia:
- Inserirem **BLOC-1** i en farem **BLOC-1***.
- Inserirem **BLOC-2**, l'explosionarem, i farem **BLOC-2*** amb els atributs N i P típics i les tríades de punts representatives dels atípics que en resultin.

En la segona inserció intermèdia:
- Inserirem **BLOC-1***.
- Inserirem **BLOC-2*** i l'explosionarem.
- Amb la inserció de **BLOC-1*** i els atributs resultants de l'explosió farem **BLOC****.
- Amb les tríades de punts en farem **BLOC-2****.

En la inserció final:
- Inserirem **BLOC****, assignant valors als atributs típics.
- Inserirem **BLOC-2****, l'explosionarem i, per a cada tríada de punts $\mathbf{O_n}$, $\mathbf{X_n}$ i $\mathbf{Y_n}$, encaixarem el bloc de suport ATRIBATIP_<**N**>_DE_<**BLOC**> en el paral·lelogram que aquests punts determinen.
- Esborrarem els punts resultants d'explosionar la inserció de **BLOC-2****.

Hem d'advertir al lector que, per haver primat en el codi l'economia de mitjans (que sovint ens ha fet caure en la mala pràctica d'aprofitar la mateixa variable per a usos diferents al llarg del programa) i en el text expositiu una claredat no sempre aconseguida, de vegades la denominació d'una mateixa realitat difereix en ambdós contextos. Per exemple, en el codi només usem dues variables, **BLOC** i **BLOC***, per representar tres noms de bloc diferents que aquí sí que distingim: **BLOC**, **BLOC*** i **BLOC****. Però encara hi ha un cas més sibil·lí i que ens podria induir a error. Ja haureu vist que, per designar el bloc auxiliar que complementa **BLOC**** com a esquelet de la inserció separada dels atributs atípics, els blocs 2D que actuen de portadors d'aquests i els seus derivats genèrics de primer grau, hem usat aquí una notació híbrida que barreja textos constants, variables de text i metallenguatge: ATRIBSATIPS_DE_<nom_de_**BLOC****>, ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**> i ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***>. Doncs bé: si és veritat que el text que correspondria a la terminació <nom_de_**BLOC****> del primer nom de bloc coincideix amb el nom **BLOC**** del genèric de segon grau a qui complementa, i que la terminació <nom_de_**BLOC**> del segon coincideix amb el nom **BLOC** del bloc original, no ho és pas que la terminació <nom_de_**BLOC***> del tercer tingui res a veure amb el nom del genèric de primer grau **BLOC*** que aquí només tindrà existència en el cas **NORM-Z** (paral·lelepípede d'encaix recte), perquè en el cas més general es desdobla en **BLOC-1*** i **BLOC-2***. Només era un recurs per suggerir que el tercer és un genèric del segon, sense recórrer a un apel·latiu tan poca-solta com <ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**>***> o alguna cosa per l'estil: penseu que no sols cada **BLOC**** i cada ATRIBSATIPS_DE_<nom_de_**BLOC****> que derivin d'uns determinats **BLOC-1*** i **BLOC-2*** seran diferents sinó que les seves insercions també ho seran. Per a cada ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**>, als efectes de crear un ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***> nou o d'aprofitar-ne un d'existent només comptarà en quin paral·lelogram hem d'encaixar el quadrat $\mathbf{1 \times 1}$.

Primer de tot haurem de revisar **SEGR-ATRIBS**, on caldrà repartir els atributs entre dues llistes **OO-2** i **OO-4**, segons que siguin típics o atípics, i fer-ne una de nova **OO-3**, on per cada atribut atípic hi guardarem els 3 punts que en l'última fase del procés ens permetran de realitzar l'encaix del bloc portador. De moment doncs, ens centrarem en aquesta funció i en les requerides des d'ella, tret de **REDEF-BLOC*S**. En aquesta primera aproximació hem mantingut les condicions d'accés en profunditat de les versions precedents i, a banda de nous elements en joc, ja hi trobareu dues simplificacions: la desaparició de la variable ***EDITATRIBS-P***, amb la incorporació definitiva dels atributs Predefinits a **BLOC-2** (conjuntament amb els Normals), que havíem anunciat, i la supressió de la característica *Verificable* en tots els que

la tinguin, orientada sobretot a unificar en nombre d'arguments la inserció dels portadors 2D d'atributs atípics. Així, en complicar-se'ns les coses per una banda (reordenació de la llista de valors d'atribut **WWAA**, agrupant primer els d'atributs típics **WWAA-1** i desplaçant cap enrera els dels atípics **WWAA-2**) i mirar per l'altra de compensar-ho reduint l'amplitud de la casuística, hem resolt un problema que arrossegàvem des de la VERSIÓ 2 i que sortia en totes les que no sol·licitaven a l'usuari els valors d'assignació als atributs en temps real d'execució de l'ordre ***INSERT***, sinó que només ho simulaven perquè la demanda s'havia produït prèviament: la repetició del missatge *Verificar valores de atributos*, quan **BLOC** tenia atributs N verificables. I, si en la VERSIÓ 16+ no haguéssim pres la iniciativa d'executar **VAL-ATRIBS** abans de **SEGR-ATRIBS** i no després (modificació que ens obliga a furgar en **BLOC** en lloc de processar directament tots els elements de **BLOC-2**, com en les versions anteriors a la 8, tot i que la primera beneficiària de la substitució de **BLOC** per **BLOC-1** i **BLOC-2** era en realitat la VERSIÓ 10), l'hauríem de prendre ara: no pas per qüestions formals (l'eficàcia del dispositiu ***DESHACER I/F*** malgrat haver polsat determinades tecles) sinó perquè **SEGR-ATRIBS** pugui disposar ja de la llista **WWAA** de valors assignats i, en paral·lel a la redistribució d'atributs P i N entre les llistes **OO-2** i **OO-4**, segons que siguin típics o atípics, pugui redistribuir aquests valors entre **WWAA-1** i **WWAA-2**.

```
(defun PUNT (P) (list '(0 . "POINT") (cons 10 (trans P VZ 0)) (cons 210 VZ)))

(defun SEGR-ATRIBS (/ BL1EX BL2EX AT AT-CP ICVP C-10 C-11 C-72 C-74 OO-3 OZ VZ)
  (setq BLOC-1 (strcat BLOC "_SENSE_ATRIBS") BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC) BL2EX (tblsearch "BLOCK" BLOC-2))
  (if (and BL1EX BL2EX)
      (setq LE (entget (cdr (assoc -2 BL2EX)))
            OO-2 (/= (cdr (assoc 0 LE)) "ENDBLK")
            OO-4 (tblsearch "BLOCK" (strcat "ATRIBATIP_1_DE_" BLOC)))
      (progn
        (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (setq ICVP (cdr (assoc 70 LE))) 2) 0))
                   (progn
                     (setq C-72 (cdr (assoc 72 LE)) C-74 (cdr (assoc 74 LE))
                           LE (if (= (logand ICVP 4) 4)
                                  (subst (cons 70 (- ICVP 4)) (assoc 70 LE) LE)
                                  LE)
                           LE (if (= C-72 4)
                                  (subst (cons 72 (setq C-72 1))
                                         (cons 72 4)
                                         (subst (cons 74 (setq C-74 2))
                                                (assoc 74 LE)
                                                LE))
                                  LE)
                           LE (if (and (< C-72 3) (> C-74 0))
                                  (subst (cons 74 0)
                                         (assoc 74 LE)
                                         (subst (cons 11 (LINIA-BASE))
;;                                                (assoc 11 LE)
;;                                                LE))
;;   Podeu usar les 3 línies precedents com alternativa a les 5 línies subsegüents
                                         (if (> C-72 0)                    ;;
                                             (subst (cons 11 (LINIA-BASE)) ;;
                                                    (assoc 11 LE)          ;;
                                                    LE)                    ;;
                                             LE))                          ;;
                                  LE)
                           OZ (last (assoc 10 LE))
                           VZ (cdr (assoc 210 LE)))
                     (if (and (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))
                         (progn
                           (setq OO-2 (cons LE OO-2))
                           (if (= (logand (cdr (assoc 70 LE)) 10) 0)
                               (setq WWAA-1 (append WWAA-1 (list (car WWAA)))
                                     WWAA (cdr WWAA))))
```

```
                        (progn
                          (if (= (logand (cdr (assoc 70 LE)) 10) 0)
                              (setq WWAA-2 (append WWAA-2 (list (car WWAA)))
                                    WWAA (cdr WWAA)))
                          (setq OO-3 (cons (PUNT (list 0 0 OZ)) OO-3)
                                OO-3 (cons (PUNT (list 1 0 OZ)) OO-3)
                                OO-3 (cons (PUNT (list 0 1 OZ)) OO-3)
                                C-10 (assoc 10 LE) C-11 (assoc 11 LE)
                                OO-4 (cons (subst (list 10 (cadr C-10)
                                                        (caddr C-10) 0)
                                                  C-10
                                                  (subst (list 11 (cadr C-11)
                                                               (caddr C-11) 0)
                                                         C-11
                                                         (subst '(210 0 0 1)
                                                                (assoc 210 LE)
                                                                LE)))
                                           OO-4)))))
                    (setq OO-1 (cons LE OO-1) AT-CP (if AT-CP T AT)))
                (setq E (entnext E)))
        (tblnext "BLOCK" T)
        (while (and (setq LB (tblnext "BLOCK"))
                    (not (setq BL*EX (wcmatch (cdr (assoc 2 LB)) N)))))
        (command "REGENT"
                 "CECOLOR" "PORCAPA"
                 "CELTYPE" "PORCAPA"
                 "CELWEIGHT" -1)
        (MAKEBLOC BLOC-1 AT-CP OO-1)
        (MAKEBLOC BLOC-2 OO-2 (append OO-3 OO-2))
        (setq J 0)
        (foreach O (reverse OO-4)
          (MAKEBLOC (strcat "ATRIBATIP_" (itoa (setq J (1+ J))) "_DE_" BLOC) T
                    (list O)))
        (if BL1EX (REDEF-BLOC*S))
        (setvar "CECOLOR" COL)
        (setvar "CELTYPE" TLIN)
        (setvar "CELWEIGHT" GLIN))))
```

Però no n'hi haurà prou amb això, perquè ens estàvem oblidant de les insercions simples: si inserim amb ***INSERT*** un bloc 3D proveït d'atributs atípics, ja hem dit abans que els punts d'ancoratge se situaràn correctament però que aquest atributs adoptaran l'orientació dels típics. Si no volem que sigui així recorrerem a ***INS3D***, sobre la qual, doncs, haurà de recaure no només la responsabilitat d'assolir unes insercions múltiples (obliqües) que integrin aquests elements amb una geometria correcta (encara que no ho facin en qualitat de components d'un mateix bloc), sinó de reixir igualment si les insercions són simples (ortogonals). En aquest supòsit, no necessitem substituir **BLOC** per **BLOC-1** (format pels elements de la llista **OO-1**) i **BLOC-2** (format pels elements de **OO-2** i **OO-3**), sinó per **NLOC-1** (**BLOC** desproveït d'atributs atípics, és a dir, format pels elements de les llistes **OO-1** i **OO-2**) i **NLOC-2** (format pels punts de la llista **OO-3**), a més dels blocs 2D portadors dels atributs atípics (formats per cadascun del elements de **OO-4**) que utilitzarem en ambdós casos. Com que caldrà evidenciar-ne la composició diferent, en comptes de <nom_de_**BLOC**>_SENSE_ATRIBS, **NLOC-1** s'anomenarà <nom_de_**BLOC**>_AMB_ATRIBSTIPS, i en lloc de ATRIBS_DE_<nom_de_**BLOC**>, **NLOC-2** s'anomenarà ATRIBSATIPS_DE_<nom_de_**BLOC**>, sense sufix (a diferència dels genèrics, també constituïts per punts, que es creen durant el procés i que, segons es produeixi o no la circumstància **NORM-Z** anomenem ATRIBSATIPS_DE_<nom_de_**BLOC***> o ATRIBSATIPS_DE_<nom_de_**BLOC****>).

La dualitat de tàndems substituts (**BLOC-1**/**BLOC-2** i **NLOC-1**/**NLOC-2**), introduïda per la irrupció en escena dels atributs atípics, multiplica la diversitat de casos que es poden presentar i de decisions que caldrà prendre. Si anomenem **NORM** l'aplec de les circumstàncies **NORM-XY** i **NORM-Z** (és a dir, l'ortogonalitat de la inserció), i **NL1EX** i **NL2EX** les certificacions d'existència de **NLOC-1** i **NLOC-2**, aquella senzilla condició **(and BL1EX BL2EX)** que fins ara definia la frontera entre l'homologació de **BLOC** (als efectes de ser tractat per **INSERT***) i la necessitat d'una exhaustiva anàlisi per assolir aquesta homologació, se'ns haurà quedat petita. Ara caldrà que

```
(or (and NORM NL1EX NL2EX (or (and BL1EX BL2EX) (not (or BL1EX BL2EX))))
    (and (not NORM) BL1EX BL2EX (or (and NL1EX NL2EX) (not (or NL1EX NL2EX)))))
```

condició que, emulant l'inexistent operador ***eXclusive OR***, podem reduir a

```
(or (and NORM NL1EX NL2EX (not (XOR BL1EX BL2EX)))
    (and (not NORM) BL1EX BL2EX (not (XOR NL1EX NL2EX))))
```

Com que a **SEGR-ATRIBS** i més enllà d'aquesta funció encara hi ha temes puntuals que queden foscos, abans de passar sense solució de continuïtat des de l'estratègia esbossada a la plasmació en codi de tot ***INS2D***/***INS3D***, aclarirem algunes qüestions:

- Entre els diferents supòsits que haurien de conduir a la destrucció de **BLOC-2** i/o **NLOC-2**, com a mitjà més efectiu per desencadenar la reconstrucció completa dels blocs que es reparteixen dins de **INSERTOK** la informació continguda a **BLOC**, cal que hi figuri la no correspondència numèrica entre les triades de punts que amaguen aquests blocs i la sèrie de blocs portadors d'atributs atípics que, tot començant per ATRIBATIP_1_DE_<nom_de_**BLOC**>, han d'anar numerats correlativament. De fet, si el nombre **N** de tríades de punts és inferior al de blocs portadors no ens molestarem a reconstruir l'entramat de blocs: ens limitarem a considerar els **N** primers blocs de la sèrie i n'ignorarem la resta (que fins i tot més endavant podria ser destruïda amb ***LIMPIA***, si abans n'haguéssim esborrat les insercions). La dràstica actuació esmentada al començament del paràgraf es reservarà al cas contrari: un nombre de tríades de punts superior al de blocs portadors.

- A fi i efecte que, en fer **(MAKEBLOC NLOC-2 ...)** o **(MAKEBLOC BLOC-2 () ...)** no surti el missatge *Redefiniendo el bloque "ATRIBS..."*, abans executarem

  ```
  (if NL2EX (command "LIMPIA" "B" NLOC-2 "N"))
  (if BL2EX (command "LIMPIA" "B" BLOC-2 "N"))
  ```

  expressions que sempre reixiran: recordeu que la inserció de **NLOC-2** s'explosiona (i que, després d'haver estat utilitzats, els punts resultants són esborrats) i la de **BLOC-2** també (els punts i/o els atributs típics resultants passen a formar part de **BLOC-2***), raó per la qual la seva destrucció no està condicionada a la prèvia destrucció de cap altre bloc.

- La mateixa llibertat d'actuació la tindrem amb **BLOC-2*** i també la conveniència d'una neteja prèvia, tot i que aquí el problema no és l'aparició de missatges (dins de **REDEF-BLOC*S**, és l'ordre ***BLOQUE*** qui s'encarregarà de redefinir **BLOC-2***, no la funció **entmake** executada des de **MAKEBLOC**, i a aquesta se la pot emmudir desactivant ***CMDECHO***) sinó l'acumulació d'escombraries: per fer via, **REDEF-BLOC*S** cerca els tàndems **BLOC-1***/**BLOC-2*** rastrejant el primer element; si hem eliminat **BLOC-1*** (havent eliminat **BLOC**** després d'esborrar-ne les insercions) i **BLOC-2***, probablement serà per la voluntat de prescindir d'insercions obliqües obsoletes; si només eliminem **BLOC-2*** (que és just el que farem), **BLOC-1*** li donarà a **REDEF-BLOC*S** la pista per redefinir-lo a ell i tornar a definir **BLOC-2** sobre la base d'un **BLOC** actualitzat; però si ens limitéssim a destruir els genèrics **BLOC-1*** (amb els requisits previs esmentats), els seus companys de tàndem sobreviurien com a material obsolet, barrejats entre blocs útils. Per evitar-ho, abans d'anar a **REDEF-BLOC*S** n'hi haurà prou a fer

  ```
  (command "LIMPIA" "B" (strcat BLOC-2 M "*") "N")
  ```

  Per les mateixes raons, tot el que hem dit a propòsit de **BLOC-1***/**BLOC-2*** també serà vàlid per a **BLOC**/**BLOC-2****, amb l'única precisió que **BLOC-2**** només té raó de ser quan **BLOC** té atributs atípics. Així, immediatament després de l'última expressió (i per tant, també abans d'anar a **REDEF-BLOC*S**) ens desempallegarem de l'esquelet de punts **BLOC-2****, de nom públic ATRIBSATIPS_DE_<nom_de_**BLOC****>, amb

  ```
  (if (or OO-4 J) (command "LIMPIA" "B" (strcat "ATRIBSATIPS_DE_" BL "_*") "N"))
  ```

  on **J** representa el nombre d'atributs atípics de numeració correlativa que tenim (si no n'hi ha cap, llavors **J** serà **nil**), i **OO-4** és la llista d'atributs atípics regularitzada tal com dèiem en el penúltim paràgraf.

- També a **SEGR-ATRIBS**, tot seguit de l'última expressió i just abans de redefinir els blocs 2D portadors d'atributs atípics, ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**>, eliminarem els genèrics ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***> les insercions dels quals hagin estat esborrades i els portadors que no tinguin insercions directes ni siguin base de cap genèric, tot i que en la VERSIÓ 17+ aquesta funció només estarà preparada per ocupar-se dels primers. De nou serà ***LIMPIA*** la protagonista:

  ```
  (if J (command "LIMPIA" "B" (strcat "ATRIBATIP_#_DE_" BL M
                                      ",ATRIBATIP_##_DE_" BL M
                                      ",ATRIBATIP_###_DE_" BL M) "N"
                 "LIMPIA" "B" (strcat "ATRIBATIP_#_DE_" BL
                                      ",ATRIBATIP_##_DE_" BL
                                      ",ATRIBATIP_###_DE_" BL) "N"))
  ```

Com en el penúltim paràgraf, la segona ordre ***LIMPIA*** té per objecte barrar el pas als missatges *Redefiniendo el bloque "ATRIBATIP_..."*, i alhora totes dues, com a l'últim, miren també d'evitar l'acumulació de la brossa constituïda per atributs atípics possiblement obsolets, tant pel que fa a insercions esborrades (que són les úniques a l'abast de l'ordre ***LIMPIA***) com per redefinicions de **BLOC** que hagin comportat una reducció en el nombre d'aquests elements singulars. Probablement, és del tot utòpic pensar en **BLOC**s de fins a **999** atributs atípics, però costa tan poc de preveure-ho sobre el paper que per nosaltres no quedarà.

- L'alternativa al plantejament algebraic, que com s'ha dit consisteix a recórrer de nou a **INSERT*** per assolir l'encaix de cada ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**> en el paral·lelogram definit per la tríada de punts que li correspongui, entre les que componen ATRIBSATIPS_DE_<nom_de_**BLOC**>, ATRIBSATIPS_DE_<nom_de_**BLOC***> o ATRIBSATIPS_DE_<nom_de_**BLOC****>, no la materialitzarem per recursivitat directa d'aquesta funció sinó per mitjà de la funció interposada **INSERT****, que també pot ser invocada directament des de **INSERTOK** en el cas d'inserció simple d'un bloc proveït d'atributs atípics (no parlem de l'ordre ***INSERT***, sinó de ***INS2D***/***INS3D*** amb la concurrència de les condicions **NORM-XY** i **NORM-Z**). L'argument **PRAL** de **INSERT***, que ens permetrà distingir entre la primera i la segona volta (serà **T** quan sigui reclamada directament des de **INSERTOK** i serà **nil** quan ho sigui des de **INSERT****), només serà indispensable per a la construcció dels derivats genèrics dels blocs portadors d'atributs atípics en la segona volta (que a diferència dels altres genèrics no anirà precedida per **INSPARCIAL**, en tenir un atribut P o N com a únic contingut), ja que tots els altres casos en què els fluxos d'execució difereixin estaran discriminats sense ambigüitat per **OO-4** (**OO-4** passa a **nil** en **INSERT****, entre d'altres coses per evitar la ciclicitat **INSERT** → INSERT* → INSERT****).

- En la VERSIÓ 14+, a propòsit de la funció **PASSA-ATRIB** (eliminada a les següents, tot i que dintre de **C:DESCOMPOK** reproduïem la seva executòria si ***EDITATRIBS-P*** era **nil**), explicàvem que en explosionar una inserció de bloc que dugués implícit un efecte de cisallament passaven dues coses: l'explosió es propagava a tots els nivells en què fos possible (insercions amb escalat uniforme o polilínies); els components irreductibles que haguessin patit cisallament (insercions de blocs públics esdevingudes insercions de blocs anònims, o atributs en què l'obliqüitat hagués variat) passaven a l'últim lloc en el conjunt resultant de l'explosió, tot i que en la descripció del bloc explosionat no ocupessin aquesta posició. Cal precisar que entre aquests desplaçaments cap enrera hi ha prioritats, segons el tipus d'objecte: per exemple, els blocs anònims passen darrera dels atributs amb canvi d'obliqüitat, tot i que aquest no sigui el nostre cas. Però sí que ens afectarà quan, mitjançant **INSPARCIAL** (reclamada des de **REDEF-BLOC*S** o **INSERT***), descomposem la penúltima inserció: si **NORM-Z**, serà una inserció de **BLOC-2** i es donarà la circumstància **(not COMPROBLIC)**; si **(not NORM-Z)**, ho serà de **BLOC-2*** i s'acomplirà **COMPROBLIC**; en aquestes condicions, haurem d'enviar els atributs P o N paral·lels a ***XY*** cap a una banda i les tríades de punts cap a una altra, amb cura de seleccionar bé els components perquè en el primer cas ($\mathbf{E_x = E_y = E_z}$) no es permutaran les seves posicions però en el segon ($\mathbf{E_z \neq E_x}$) sí. Per això, abans d'anar a **REST-OBLIC** (si s'escau), caldrà intervenir amb el dispositiu

```
(if (and OO-4 (or NORM-Z COMPROBLIC))
    (progn
      (setq SA (ssadd) K (if NORM-Z (- (sslength SS) (* 3 OO-4)) 1))
      (repeat (* 3 OO-4)
              (setq E (ssname SS K))
              (ssadd E SA)
              (ssdel E SS))))
```

subordinat a l'existència d'atributs atípics (si no, no hi hauria punts), que ens lliurarà un conjunt **SS** format per la inserció de **BLOC-1** o **BLOC-1*** i pels atributs típics que resulten de l'explosió de **BLOC-2** o **BLOC-2***. Sols que ara **SA** esdevindrà alguna cosa més que un conjunt de selecció auxiliar: estarà format per les tríades de punts que ens han de permetre situar els atributs atípics.

- I ja que hem citat **C:DESCOMPOK**, la possibilitat que **BLOC** tingui atributs atípics aporta dues noves dificultats (a banda de reflectir-ho en el missatge), atès que de l'aplicació de ***INS2D***/***INS3D*** en resultarà més d'una inserció:
  - D'una banda, a l'usuari (que no té perquè saber de la fragmentació en diverses insercions) tant li pot donar per fer clic sobre la inserció principal com per fer-ho sobre qualsevol dels atributs atípics, i en aquest segon cas la funció haurà de poder localitzar la inserció principal a la base de dades del dibuix, remuntant-la a partir de la inserció de bloc portador seleccionada.

- A partir de la inserció principal, **C:DESCOMPOK** haurà d'anar recorrent la base de dades (en sentit normal, contrari al citat en últim lloc) i descomponent les insercions que formin la seqüència ATRIBATIP_1_DE_..., ATRIBATIP_2_DE_..., ATRIBATIP_3_..., ... : així ho expressarem, amb la condició
  (wcmatch (cdr (assoc 2 LE)) (strcat "ATRIBATIP_" (itoa K) "_DE_*"))
  perquè pot ser que en algun cas el marc d'encaix sigui rectangular (i haguem inserit el portador mateix, ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**>) mentre que a d'altres no (i haguem inserit un derivat ATRIBATIP_ <núm.>_DE_<nom_de_**BLOC***>).

No cal dir que aquest dispositiu només funcionarà correctament si l'usuari no ha manipulat el resultat d'aplicar ***INS2D***/***INS3D***, esborrant-ne la inserció principal o la d'algun portador d'atribut atípic. I, per acabar, una observació a propòsit de com es comporta la funció **entnext** en el cas d'explosionar una inserció, sense la qual potser no s'entendrà del tot la seqüència d'instruccions

```
(wile (not (setq E (car (entsel))))) (setq LE (entget E))
.....................................
(command "DESCOMP" E ())
.............................
(setq NEXTE (if NEXTE NEXTE (entnext E)) ...)
```

Podríem suposar, en la mesura que la inserció **E** seleccionada ha deixat d'existir després de l'explosió, que **E** mantindrà un valor de nom intern d'objecte que ja no té cap presència a la base de dades del dibuix i que l'expressió **(entnext E)** donarà error, però no és així: quan **BLOC** té atributs P o N típics, executant **(entnext E)** abans d'explosionar **E** resultarà el primer atribut amb valor assignat (objecte **"ATTRIB"**, no pas **"ATTDEF"**), però fent-ho després ens trobarem amb el primer objecte situat després de la inserció explosionada, perquè els resultants de l'explosió hauran passat al final del dibuix, per darrera dels que venien a continuació; i quan **BLOC** també té atributs atípics, les insercions dels seus blocs portadors són precisament els primers d'aquests objectes.

De seguida veurem el codi al complet (tret del dispositiu ***DESHACER I*/*F***, suprimit temporalment per moure'ns més lleugers), però abans en destacarem les mancances: aquesta primera versió per a blocs amb atributs atípics és coixa, perquè únicament s'hi contemplen les aplicacions ***INS2D***/***INS3D*** que donen lloc al tàndem **BLOC-1**/**BLOC-2** (o a **NLOC-1**/**NLOC-2**) i als blocs 2D portadors d'aquests components singulars, la primera vegada que tractem amb **BLOC** o, si més no, la primera en què ho fem després d'haver eliminat **BLOC-2** (o **NLOC-2**), acceptant així que volem actualitzar la resta de blocs auxiliars per adequar-los a una eventual redefinició de **BLOC**. En **INSERT*** conservem l'estructura (heretada de la VERSIÓ 16+) que fa possible l'aprofitament dels **BLOC-1***, **BLOC-2***, **BLOC**** i **BLOC-2**** o la seva creació (segons que existeixin o no), igual que la nova funció **INSERT**** fa amb els portadors d'atributs atípics i els seus genèrics, però hi manca una informació tan indispensable com les llistes **WWAA-1** (valors per assignar als atributs típics) i **WWAA-2** (valors per als atípics) que només l'exploració en profunditat realitzada a **SEGR-ATRIBS** (i reservada als casos abans esmentats) organitza a partir de la llista única **WWAA** (on uns i altres desfilen barrejats, en l'ordre d'aparició dictat per **BLOC**). Com que l'assumpte té diverses implicacions (encara que només juguéssim amb **BLOC**s definits de manera que els atributs P i N típics se situessin abans que els atípics y fácilment poguéssim muntar **WWAA-1** i **WWAA-2** a partir de **WWAA**, només sabent quants n'hi ha dels segons, no és legítim treure sempre aquesta llista directament de **BLOC**, per las raons que s'adduiran arribat el moment), preferim aparcar-lo i reobrir-lo més endavant, amb una visió més àmplia de la problemàtica i un cop resoltes qüestions col·laterals.

```
; VERSIÓ 17+

(defun C:INS2D () (INSERTOK T))

(defun C:INS3D () (INSERTOK ()))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
```

```
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun U*V (U V)
  (list (- (* (cadr U) (last V)) (* (last U) (cadr V)))
        (- (* (last U) (car  V)) (* (car  U) (last V)))
        (- (* (car  U) (cadr V)) (* (cadr U) (car  V)))))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS (/ ICVP M VD V LLAA)
  (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
  (while E
         (if (and (= (cdr (assoc 0 (setq LE (entget E)))) "ATTDEF")
                  (= (logand (setq ICVP (cdr (assoc 70 LE))) 10) 0))
             (progn
               (if (and (not LLAA) ATREQ)
                   (prompt "\nIndique valores de atributo"))
               (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LE))
                     (cdr (assoc 3 LE)) (cdr (assoc 1 LE)))
               (setq LLAA (cons (list W N M V) LLAA))))
         (setq E (entnext E)))
  (setq W ())
  (foreach LA LLAA
    (if (car LA)
        (progn
          (if (and (not W) ATREQ) (prompt "\nVerificar valores de atributo"))
          (VATR (cons "" W) (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq WWAA (cons V WWAA))))

(defun MAKEBLOC (BLOC/2 ATRIBS OO)
  (entmake (list '(0 . "BLOCK") (cons 2 BLOC/2) '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun MNLOC ()
  (MAKEBLOC (setq BLOC NLOC-1) (or AT-CP OO-2) (append OO-2 OO-1))
  (MAKEBLOC NLOC-2 () OO-3))

(defun MBLOC ()
  (MAKEBLOC BLOC-1 AT-CP OO-1)
  (MAKEBLOC BLOC-2 OO-2 (append OO-3 OO-2)))

(defun LINIA-BASE ()
  (polar (cdr (assoc 11 LE))
         (- (cdr (assoc 50 LE)) PI/2)
         (* (cdr (assoc 40 LE))
            (if (= C-74 1)
                (/ 1.0 -3)
                (if (= C-74 2) 0.5 1)))))
```

```
(defun PUNT (P) (list '(0 . "POINT") (cons 10 (trans P VZ 0)) (cons 210 VZ)))

(defun REST-OBLIC ()
  (setq A (cos (cdr (assoc 51 (entget E*)))))
  (if (not (equal A 1 Q0))
      (progn
        (setq LE (entget E)
              LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A)) (assoc 40 LE) LE)
              LE (subst (cons 41 (* (cdr (assoc 41 LE)) A)) (assoc 41 LE) LE))
        (entmod LE))))

(defun INSPARCIAL (EX EY EZ ANG / COMPROBLIC)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast))
        COMPROBLIC (not (= EX EY EZ)))
  (if (or OO-2 OO-4)
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast))
        (setq SA (ssget "P") K -1)
        (while (setq E (ssname SA (setq K (1+ K))))
               (ssadd E SS))))
  (if (and OO-4 (or NORM-Z COMPROBLIC))
      (progn
        (setq SA (ssadd) K (if NORM-Z (- (sslength SS) (* 3 OO-4)) 1))
        (repeat (* 3 OO-4)
                (setq E (ssname SS K))
                (ssadd E SA)
                (ssdel E SS))))
  (if COMPROBLIC
      (progn
        (setq K 0 E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC-2))))
        (while (setq K (1+ K) E (ssname SS K))
               (REST-OBLIC)
               (setq E* (entnext E*))))))

(defun XOR (A B) (and (not (and A B)) (or A B)))

(defun REDEF-BLOC*S (/ LB NZ)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (if BL1EX
      (progn
        (setq J (1+ (strlen BLOC-1))
              NZ NORM-Z NORM-Z ())
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC-1* (cdr (assoc 2 LB)))
                            (strcat BLOC-1 M (substr M 2)))
                   (progn
                     (setq E (cdr (assoc -2 LB))
                           LE (entget E)
                           K (cdr (assoc 41 LE))
                           BLOC-2* (strcat BLOC-2 (substr BLOC-1* J)))
                     (command "SCP" "EZ" O (cdr (assoc 210 LE)))
                     (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (command "SCP" "PR")
                     (PRO-DESHACER-I/F))))
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat N "*"))
                   (progn
```

```
                    (setq E (cdr (assoc -2 LB))
                          LE (entget E)
                          BLOC-1 (cdr (assoc 2 LE))
                          E (substr BLOC-1 J) NORM-Z (= E "")
                          BLOC-2 (strcat BLOC-2* E))
                    (if (not NORM-Z) (command "SCP" "EZ" O (cdr (assoc 210 LE))))
                    (INSPARCIAL (cdr (assoc 41 LE))
                                (cdr (assoc 42 LE))
                                (cdr (assoc 43 LE))
                                (/ (* (cdr (assoc 50 LE)) 180) PI))
                    (if (not NORM-Z) (command "SCP" "PR"))
                    (command "BLOQUE" BLOC* O SS "")
                    (if OO-4 (command "BLOQUE" (strcat "ATRIBSATIPS_DE_" BLOC*)
                                                       O SA "")))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* NORM-Z NZ)))
  (setvar "CLAYER" CAPA))

(defun J-PUNTS (LB)
  (setq E (cdr (assoc -2 LB)) J 0)
  (while E
         (setq J (if (= (cdr (assoc 0 (entget E))) "POINT") (1+ J) J)
               E (entnext E)))
  J)

(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 NL1EX NL2EX BL1EX BL2EX AT AT-CP
                      ICVP C-10 C-11 C-72 C-74 OO-3 OZ VZ)
  (setq NORM (and NORM-XY NORM-Z)
        NLOC-1 (strcat BLOC "_AMB_ATRIBSTIPS")
        NL1EX (tblsearch "BLOCK" NLOC-1)
        NLOC-2 (strcat "ATRIBSATIPS_DE_" BLOC)
        NL2EX (tblsearch "BLOCK" NLOC-2)
        BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2) K 0
        OO-4 (while (tblsearch "BLOCK"
                               (strcat "ATRIBATIP_" (itoa (setq K (1+ K))) "_DE_"
                                       BLOC))
                    K)
        K (if (and NL2EX BL2EX)
              (if (= (J-PUNTS NL2EX) (J-PUNTS BL2EX)) J)
              (if NL2EX
                  (J-PUNTS NL2EX)
                  (if BL2EX (J-PUNTS BL2EX))))
        J OO-4)
  (if K
      (if (< K (if OO-4 (* 3 OO-4) 0))
          (setq OO-4 (/ K 3))
          (if (> K (if OO-4 (* 3 OO-4) 0))
              (progn
                (setq NL2EX () BL2EX ())
                (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N")))))
  (if (or (and NORM NL1EX NL2EX (not (XOR BL1EX BL2EX)))
          (and (not NORM) BL1EX BL2EX (not (XOR NL1EX NL2EX))))
      (setq OO-2 (if BL2EX (/= (cdr (assoc 0 (entget (cdr (assoc -2 BL2EX)))))
                               "ENDBLK")))
      (progn
        (setq OO-4 () E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (setq ICVP (cdr (assoc 70 LE))) 2) 0))
                   (progn
                     (setq C-72 (cdr (assoc 72 LE))
                           C-74 (cdr (assoc 74 LE))
                           LE (if (= (logand ICVP 4) 4)
                                  (subst (cons 70 (- ICVP 4)) (assoc 70 LE) LE)
                                  LE)
```

```
                              LE (if (= C-72 4)
                                     (subst (cons 72 (setq C-72 1))
                                            (cons 72 4)
                                            (subst (cons 74 (setq C-74 2))
                                                   (assoc 74 LE)
                                                   LE))
                                     LE)
                              LE (if (and (< C-72 3) (> C-74 0))
                                     (subst (cons 74 0)
                                            (assoc 74 LE)
;;                                          (subst (cons 11 (LINIA-BASE))
;;                                                 (assoc 11 LE)
;;                                                 LE))
;;   Podeu usar les 3 línies precedents com alternativa a les 5 línies subsegüents
                                          (if (> C-72 0)                          ;;
                                              (subst (cons 11 (LINIA-BASE))       ;;
                                                     (assoc 11 LE)                ;;
                                                     LE)                          ;;
                                              LE))                                ;;
                                     LE)
                              OZ (last (assoc 10 LE))
                              VZ (cdr (assoc 210 LE)))
                         (if (and (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))
                             (progn
                               (setq OO-2 (cons LE OO-2))
                               (if (= (logand (cdr (assoc 70 LE)) 10) 0)
                                   (setq WWAA-1 (append WWAA-1 (list (car WWAA)))
                                         WWAA (cdr WWAA))))
                             (progn
                               (if (= (logand (cdr (assoc 70 LE)) 10) 0)
                                   (setq WWAA-2 (append WWAA-2 (list (car WWAA)))
                                         WWAA (cdr WWAA)))
                               (setq OO-3 (cons (PUNT (list 0 0 OZ)) OO-3)
                                     OO-3 (cons (PUNT (list 1 0 OZ)) OO-3)
                                     OO-3 (cons (PUNT (list 0 1 OZ)) OO-3)
                                     C-10 (assoc 10 LE) C-11 (assoc 11 LE)
                                     OO-4 (cons (subst (list 10 (cadr C-10)
                                                             (caddr C-10) 0)
                                                       C-10
                                                       (subst (list 11 (cadr C-11)
                                                                    (caddr C-11) 0)
                                                              C-11
                                                              (subst '(210 0 0 1)
                                                                     (assoc 210 LE)
                                                                     LE)))
                                                OO-4)))))
                        (setq OO-1 (cons LE OO-1) AT-CP (if AT-CP T AT)))
                  (setq E (entnext E)))
            (if (or OO-4 (not NORM))
                (progn
                  (command "REGENT"
                           "CECOLOR" "PORCAPA"
                           "CELTYPE" "PORCAPA"
                           "CELWEIGHT" -1)
                  (if (and NL1EX NL2EX (not BL1EX) (not BL2EX))
                      (MBLOC)
                      (if (and BL1EX BL2EX (not NL1EX) (not NL2EX))
                          (MNLOC)
                          (progn
                            (if NL2EX (command "LIMPIA" "B" NLOC-2 "N"))
                            (if BL2EX (command "LIMPIA" "B" BLOC-2 "N"))
                            (if (or NL1EX NL2EX NORM) (MNLOC))
                            (if (or BL1EX BL2EX (not NORM)) (MBLOC))
                            (command "LIMPIA" "B" (strcat BLOC-2 M "*") "N")
                            (if (or OO-4 J)
                                (command "LIMPIA" "B"
                                                  (strcat "ATRIBSATIPS_DE_" BL "_*")
                                                  "N"))
```

```
                        (if J
                            (command "LIMPIA" "B"
                                              (strcat "ATRIBATIP_#_DE_" BL M
                                                      ",ATRIBATIP_##_DE_" BL M
                                                      ",ATRIBATIP_###_DE_" BL M)
                                              "N"
                                     "LIMPIA" "B"
                                              (strcat "ATRIBATIP_#_DE_" BL
                                                      ",ATRIBATIP_##_DE_" BL
                                                      ",ATRIBATIP_###_DE_" BL)
                                              "N"))
                        (if OO-4
                            (progn
                              (setq J 0)
                              (foreach O (reverse OO-4)
                                (MAKEBLOC (strcat "ATRIBATIP_"
                                                  (itoa (setq J (1+ J)))
                                                  "_DE_" BL) T (list O)))
                              (setq OO-4 (length OO-4))))
                        (if (or BL1EX OO-4) (REDEF-BLOC*S)))))
                (if (not (atom OO-4)) (setq OO-4 (length OO-4)))
                (setvar "CECOLOR" COL)
                (setvar "CELTYPE" TLIN)
                (setvar "CELWEIGHT" GLIN))))))

(defun CALCULA (U V OV AGUT U* U** OU* OU** V**)
  (setq A (polar O (- (angle O V) PI/2) OV)
        W (if AGUT (angle A U) (angle U A))
        B (mapcar '/ (mapcar '+ A U) '(2 2 2))
        W (angle O (polar B W (distance O B))))
  (set U* (inters U (polar U W 1) O B ()))
  (set U** (inters O (polar O (+ W PI/2) 1) U (eval U*) ()))
  (set OU* (distance O (eval U*)))
  (set OU** (distance O (eval U**)))
  (set V** (inters V (polar V W 1) O (eval U**) ())))

(defun FIX+ (O / N)
  (setq N (itoa (if (> (- O (setq N (fix O))) 0.5) (1+ N) N)))
  (repeat (- (strlen M) 1 (strlen N)) (setq N (strcat "0" N)))
  N)

 (defun INSERT** (BLOC X Y Z G / JJ KK SSAA)
  (command "INSERT" (strcat "ATRIBSATIPS_DE_" BLOC) O "XYZ" X Y Z G
           "DESCOMP" (entlast)
           "SCP" "")
  (setq SSAA (ssget "P") JJ 0 KK -1)
  (while (setq JJ (1+ JJ) BLOC (strcat "ATRIBATIP_" (itoa JJ) "_DE_" BL)
               LE (tblsearch "BLOCK" BLOC))
         (setq LE (entget (cdr (assoc -2 LE)))
               WA (if (= (logand (cdr (assoc 70 LE)) 10) 0) (car WWAA-2))
               WWAA-2 (if WA (cdr WWAA-2))
               KK (1+ KK) O (cdr (assoc 10 (entget (ssname SSAA KK))))
               KK (1+ KK) X (cdr (assoc 10 (entget (ssname SSAA KK))))
               KK (1+ KK) Y (cdr (assoc 10 (entget (ssname SSAA KK)))))
         (command "SCP" "3" O X Y)
         (setq OX (distance O X) OY (distance O Y) OZ OX
               O '(0 0 0)
               X (list OX 0 0) Y (trans Y 0 1) Z (U*V X Y)
               Z-XY O
               I (if (< (last Z) 0) -1 1)
               NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                              (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
               NORM-Z T OO-4 () WWAA-1 (list WA))
         (if NORM-XY
             (eval (append '(command "INSERT" BLOC O OX OY 0)
                           (if ATREQ WWAA-1)))
             (INSERT* ()))
         (command "SCP" "PR"))
```

```
  (command "SCP" "PR"
           "BORRA" SSAA ""))

(defun INSERT* (PRAL / X* X** Y* Y** Z** OX* OX** Z-XY* XY XY* XY** OXY* OXY**
                       SA SS ABLOC*)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X
            Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'OX* 'OX** 'Y**)
        (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
              A (inters Z-XY (polar Z-XY W 1) O X** ())
              Z-XY* (polar A (angle A Z-XY)
                           (/ (* (distance A Z-XY) (distance X* X**))
                              (distance X X**))))))
  (if NORM-Z
      (setq BLOC* (strcat BLOC "_"))
      (progn
        (setq X (inters X* Y* O Z-XY* ()) X (if X X Z-XY*)
              Z-XY (- (angle O X) (angle Y* X*))
              Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
              Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
              Z (trans (list (car Z-XY*) (cadr Z-XY*) (last Z)) 1 0))
        (command "SCP" "Z" O X
                 "SCP" "X" (* I 90))
        (setq XY (list OX* 0 0)
              Z (trans Z 0 1))
        (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'OXY* 'OXY** 'Z**)
        (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                            (FIX+ (/ OXY** OXY* Q0)))
              BLOC* (strcat BLOC K)
              BLOC-1* (strcat BLOC-1 K)
              BLOC-2* (strcat BLOC-2 K)
              BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
        (command "SCP" "PR" "SCP" "PR")
        (if (not BL*EX)
            (progn
              (INSPARCIAL (setq K (/ OXY* OXY**)) K K X*)
              (command "SCP" "Z" O X
                       "SCP" "X" 90
                       "GIRA" SS "" O XY*
                       "SCP" "Z" O XY**
                       "BLOQUE" BLOC-1* O (setq E (ssname SS 0)) ""
                       "BLOQUE" BLOC-2* O (ssdel E SS) ""
                       "SCP" "PR")))
        (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*
              BLOC* (strcat BLOC* "_" (FIX+ (/ OXY** OX* Q0))))))
  (if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
  (if OO-4 (setq ABLOC* (strcat "ATRIBSATIPS_DE_" BLOC*)))
  (if (or NORM-Z BL*EX) (setq BL*EX (and (tblsearch "BLOCK" BLOC*)
                                         (if OO-4 (tblsearch "BLOCK" ABLOC*) T))))
  (if (not BL*EX)
      (progn
        (if NORM-Z
            (progn
              (setq K (/ OX* OX**))
              (if PRAL
                  (INSPARCIAL K K K X*)
                  (command "ATTREQ" 0
                           "INSERT" BLOC O K K X*
                           "ATTREQ" (if ATREQ 1 0)
                           "DESCOMP" (entlast))))
```

```
            (progn
              (if B (command "SCP" "Z" O X
                             "SCP" "X" 90))
              (INSPARCIAL (setq J (/ 1 OX**) K (* OXY** J))
                          (* (distance Z Z**) J) (* K (/ OX* OXY*)) XY**)
              (command "SCP" "PR" "SCP" "PR")))
        (command "SCP" "Z" O X**
                 "BLOQUE" BLOC* O (if PRAL SS "P") "")
        (if OO-4 (command "BLOQUE" ABLOC* O SA ""))
        (command "SCP" "PR")))
  (command "CELWEIGHT" GLIN
           "CELTYPE" TLIN
           "CECOLOR" COL
           "CLAYER" CAPA)
  (eval (append '(command "INSERT" BLOC* O "XYZ" OX** (setq J (distance Y Y**))
                          (setq K (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1)
                                     OX** I)) X**)
                (if ATREQ WWAA-1)))
  (if OO-4 (INSERT** BLOC* OX** J K X**)))

(defun INSERTOK (2D / Q0 2*PI PI/2 M N CAPA COL TLIN GLIN ECO OSN ATDIA ATREQ
                      EXPERT BL BLOC BLOC* BLOC-1 BLOC-2 BLOC-1* BLOC-2* BL*EX
                      WWAA WWAA-1 WWAA-2 WA O X Y Z UV Z-XY OX OY OZ X-Y X-O-Y
                      A B E LE E* I J K W NORM-XY NORM-Z OO-1 OO-2 OO-4)
  (setq Q0 0.001 M "_")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq M (strcat M "#")))
  (setq 2*PI (* PI 2)
        PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        EXPERT (getvar "EXPERT")
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        N (strcat BLOC M) W T)
  (while W
    (setq O (getpoint "\nPrecise punto de inserción: "))
    (foreach P (if 2D '("X" "Y") '("X" "Y" "Z"))
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                  " desde el punto de inserción: "))
            W (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC  " <1>: "))
            W (if W W 1))
      (set (read P) (mapcar '(lambda (CO CA) (+ CO (/ (- CA CO) W))) O A))
      (set (read (strcat "O" P)) (/ (distance O A) W)))
    (setq OZ (if 2D OX OZ)
          W (if (equal (setq UV (U*V (mapcar '- X O) (mapcar '- Y O)))
                       '(0 0 0) Q0)
                " e Y están alineados."
                (if (and (not 2D) (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                                (mapcar '* UV (mapcar '- Z O)))
                                         0 Q0))
                    ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq Z (trans (if 2D (mapcar '+ O UV) Z) 1 0)
        Z-XY Y Y (trans Y 1 0))
```

```
  (if (= (logand (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) 2) (VAL-ATRIBS))
  (setvar "CMDECHO" 0)
  (command "OSMODE" 0
           "ATTDIA" 0
           "EXPERT" 2
           "SCP" "3" O X Z-XY)
  (setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1)
        Z-XY (list (car Z) (cadr Z) 0)
        I (if (< (last Z) 0) -1 1)
        NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                       (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
        NORM-Z (equal Z-XY O Q0)
        BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (if (and NORM-XY NORM-Z)
      (progn
        (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                      (if ATREQ (append WWAA (if OO-4 () W)))))
        (if OO-4 (INSERT** BL OX OY (* OZ I) 0)))
      (INSERT* T))
  (command "SCP" "PR"
           "INSNAME" BL
           "EXPERT" EXPERT
           "ATTDIA" ATDIA
           "OSMODE" OSN
           "CMDECHO" ECO)
  (princ))

(defun ENTPRECEDING ()
  (setq A (entnext) K A)
  (while (not (equal K E))
         (setq A K K (entnext K)))
  A)

(defun C:DESCOMPOK (/ Q0 ECO BLOC* E LE E* SA A N K NEXTE)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D (si hay atributos ")
  (prompt "con caracteres")
  (prompt "\noblicuos o no situados en su plano base, la orden DESCOMP no lo ")
  (prompt "resolverá bien)")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              BLOC* (if (wcmatch BLOC* "ATRIBATIP_#*")
                        (progn
                          (setq NEXTE E)
                          (while (or (/= (cdr (assoc 0 (setq E (ENTPRECEDING)
                                                             LE (entget E))))
                                         "INSERT")
                                     (if (wcmatch (cdr (assoc 2 LE))
                                                  "ATRIBATIP_#*")
                                         (setq NEXT E))))
                          (cdr (assoc 2 LE)))
                        BLOC*)
              ECO (getvar "CMDECHO"))
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (if (and (wcmatch BLOC* (strcat "*_" N ",*_" N N ",*_AMB_ATRIBSTIPS"))
                 (setq SA (ssget "P")))
            (progn
              (setq NEXTE (if NEXTE NEXTE (entnext E))
                    E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))) K -1)
              (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                     (REST-OBLIC))))
        (setq K 0)
```

```
      (while (and (setq E NEXTE)
                  (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                  (wcmatch (setq BLOC* (cdr (assoc 2 LE)))
                           (strcat "ATRIBATIP_" (itoa K) "_DE_*")))
             (command "DESCOMP" E ())
             (setq NEXTE (entnext E) E (ssname (ssget "P") 0)
                   E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))
             (REST-OBLIC))
      (setvar "CMDECHO" ECO))
    (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

A més de la restricció explicitada just abans de presentar el codi (quan **BLOC** té atributs atípics, la VERSIÓ 17+ només garanteix la primera aplicació ***INS2D***/***INS3D***), el lector perspicaç probablement haurà trobat a faltar a **REDEF-BLOC*S** la presència dels genèrics 2D que anomenem ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***> i que han permès l'encaix dels blocs **1 × 1** ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**>, portadors d'atributs atípics, en els paral·lelograms definits per les tríades de punts del bloc 3D ATRIBSATIPS_DE_<nom_de_**BLOC****>, i tindrà raó: de moment l'havíem deixada així, pendents d'una reestructuració en profunditat. Recordarem que, quan **BLOC** ha estat redefinit i hem destruït **BLOC-2** per forçar-ne l'actualització (no només de **BLOC-2** sinó de **BLOC-1**) en **SEGR-ATRIBS**, el procés es completa a **REDEF-BLOC*S** on, a partir del tàndem substitut redefinit, la revisió es propaga als genèrics existents (de primer a **BLOC-1*** i **BLOC-2***, i d'aquí a **BLOC*** o a **BLOC****) per tal d'actualitzar les insercions realitzades però sobretot per assegurar que, quan aquests genèrics de **BLOC** siguin utilitzats per futures aplicacions de **INSERTOK**, tot respongui a la nova definició. Ara, si **BLOC** té atributs atípics el procés s'hauria d'estendre als blocs 2D que els vehiculen, i d'aquí ve el problema: a **REDEF-BLOC*S** no recalculem la geometria dels genèrics en funció de la que volem que tinguin les insercions finals, tornant a executar **INSERTOK** (només faltaria!), sinó que aprofitem el fet que el primer component és una inserció de **BLOC-1** (bloc eventualment buit, si **BLOC** estava compost exclusivament per atributs N i P) o de **BLOC-1***, per obtenir-ne la informació relativa al ***SCP*** sota el qual s'havia realitzat la inserció, els factors d'escala i l'angle de gir; però com que els blocs portadors dels atributs atípics no tenen cap altre contingut i no han estat objecte d'una subdivisió anàloga a la de **BLOC** en **BLOC-1** i **BLOC-2**, no podrem treure d'enlloc aquestes dades.

De tota manera, precisem què està realment en joc: si la redefinició de **BLOC** afecta atributs P o N, ja havíem dit que no hi havia res a pelar pel que feia a les insercions realitzades, i no parlem només dels continguts (que ja és lògic que es mantinguin en la seva diversitat) sinó de posició, inclinació i de totes les característiques lligades a l'estil de text; si aquests atributs són atípics, a més, haurem de considerar que el pla de l'atribut és una altra característica irrecuperable pel que fa a les insercions realitzades, cosa que fins i tot pot comportar que un atribut situat en un objecte pla (posem una cara 3D) quedi situat a fora (a fora del pla i/o del contorn d'aquesta cara), si la redefinició de **BLOC** ha afectat components de **BLOC-1**. I, quan això sigui inassumible, no hi haurà més remei que esborrar les insercions prèvies i repetir-les amb el nou **BLOC**. Convenia aclarir això per no crear falses expectatives sobre la VERSIÓ 18+: només pretenem que les insercions que es realitzin a partir del moment en què la redefició de **BLOC** sigui reconeguda per ***INS2D***/***INS3D*** (mitjançant la destrucció intencionada dels blocs **BLOC-2** o **NLOC-2**) responguin als nous continguts, atributs atípics inclosos. Si renunciem a que redefinicions de **BLOC** amb incidència sobre els atributs atípics afectin insercions obliqües futures quan depenguin de genèrics **BLOC**** anteriors al canvi (ens referim a casos de **BLOC**** i ATRIBSATIPS_DE_<nom_de_**BLOC****> apareguts abans que **BLOC** sigui redefinit, en què no tots els atributs atípics renovaran el seu aspecte, tot i l'actualització dels genèrics esmentats[*]), ja en tindríem prou

[*] En realitar, la no actualització es refereix als genèrics 2D ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***> previs a la redefinició de **BLOC**, que són els grans absents de **REDEF-BLOC*S**. Així, es pot donar el cas d'algun **BLOC**** anterior a la redefinició en què, en una inserció posterior, els punts de la tríada associada de ATRIBSATIPS_DE_<nom_de_**BLOC****> formin angle recte i això permeti d'inserir directament ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**>, que sí que s'haurà renovat, o perquè defineixin un paral·lelogram que precisa del concurs d'un genèric 2D ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***> diferent dels existents i que per tant es crearà segons la nova definició de **BLOC**. Com també es pot donar el cas d'uns **BLOC**** i ATRIBSATIPS_DE_<nom_de_**BLOC****> creats després de ser redefinit **BLOC** però que, en ser inserits amb uns determinats factors d'escala, el segon doni lloc a una tríada que admet el concurs d'un genèric 2D ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***> preexistent a la redefinició de **BLOC** i per tant no renovat. Heu de tenir present el que dèiem abans a propòsit de la poc afortunada notació que estem utilitzant aquí.

amb una VERSIÓ 17+ ampliada per donar també cabuda a les insercions no inaugurals (recordem que en això està coixa). Si no, caldrà introduir certs canvis (orientats exclusivament a la superació d'aquest problema, perquè encara la deixen coixa del mateix peu), que ara presentem i tot seguit justificarem:

- Cap al final de **REDEF-BLOC*S**, abans de restaurar el valor de la variable ***CLAYER***, inserirem l'expressió:

```
(if OO-4
    (progn
      (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2 J 0)
      (repeat OO-4
              (setq J (1+ J) BLOC-2 (strcat "ATRIBATIP_" (itoa J) "_DE_" BL))
              (tblnext "BLOCK" T)
              (while (setq LB (tblnext "BLOCK"))
                     (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                                  (strcat BLOC-2 M))
                         (progn
                           (setq BLOC-1 "BLOC_NUL"
                                 E (cdr (assoc -2 LB))
                                 LE (entget E)
                                 K (cdr (assoc 41 LE)))
                           (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                           (command "BLOQUE" BLOC* O SS "")))))
      (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2*)))
```

- Cap al final de **SEGR-ATRIBS**, a l'expressió

```
(if OO-4
    (progn
      (setq J 0)
      (foreach O (reverse OO-4)
        (MAKEBLOC (strcat "ATRIBATIP_"
                          (itoa (setq J (1+ J)))
                          "_DE_" BL) T (list O)))
      (setq OO-4 (length OO-4))))
```

  hi garantirem l'existència d'un bloc buit, i quedarà així:

```
(if OO-4
    (progn
      (if (not (tblsearch "BLOCK" "BLOC_NUL"))
          (MAKEBLOC "BLOC_NUL" () ()))
      (setq J 0)
      (foreach O (reverse OO-4)
        (MAKEBLOC (strcat "ATRIBATIP_"
                          (itoa (setq J (1+ J)))
                          "_DE_" BL) T (list O)))
      (setq OO-4 (length OO-4))))
```

- Suprimirem l'argument **PRAL** de la definició de la funció **INSERT***, canvi que afectarà:
  - Les crides a aquesta funció des de **INSERTOK** (**(INSERT* T)**) i des de **INSERT**** (**(INSERT* ())**), que quedaran en **(INSERT*)**.
  - El contingut de **INSERT**** on, poc abans d'aquesta última crida afegirem les assignacions subratllades

```
(while (setq JJ (1+ JJ) BLOC (strcat "ATRIBATIP_" (itoa JJ) "_DE_" BL)
             LE (tblsearch "BLOCK" BLOC))
       (setq LE (entget (cdr (assoc -2 LE)))
             WA (if (= (logand (cdr (assoc 70 LE)) 10) 0) (car WWAA-2))
             WWAA-2 (if WA (cdr WWAA-2))
             KK (1+ KK) O (cdr (assoc 10 (entget (ssname SSAA KK))))
             KK (1+ KK) X (cdr (assoc 10 (entget (ssname SSAA KK))))
             KK (1+ KK) Y (cdr (assoc 10 (entget (ssname SSAA KK)))))
       (command "SCP" "3" O X Y)
       (setq OX (distance O X) OY (distance O Y) OZ OX
             O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (U*V X Y)
             Z-XY O I (if (< (last Z) 0) -1 1)
             BLOC-1 "BLOC_NUL" BLOC-2 BLOC
             NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                            (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
             NORM-Z T OO-2 T OO-4 () WWAA-1 (if WA (list WA)))
```

```
          (if NORM-XY
              (eval (append '(command "INSERT" BLOC O OX OY 0)
                            (if ATREQ WWAA-1)))
              (INSERT*))
          (command "SCP" "PR"))
```

- Com a conseqüència d'aquestes assignacions, el contingut de **INSERT***, en què l'expressió

```
(if (not BL*EX)
    (progn
      (if NORM-Z
          (progn
            (setq K (/ OX* OX**))
            (if PRAL
                (INSPARCIAL K K K X*)
                (command "ATTREQ" 0
                         "INSERT" BLOC O K K X*
                         "ATTREQ" (if ATREQ 1 0)
                         "DESCOMP" (entlast))))
          (progn
            (if B (command "SCP" "Z" O X
                           "SCP" "X" 90))
            (INSPARCIAL (setq J (/ 1 OX**) K (* OXY** J))
                        (* (distance Z Z**) J) (* K (/ OX* OXY*)) XY**)
            (command "SCP" "PR" "SCP" "PR")))
      (command "SCP" "Z" O X**
               "BLOQUE" BLOC* O (if PRAL SS "P") "")
      (if OO-4 (command "BLOQUE" ABLOC* O SA ""))
      (command "SCP" "PR")))
```

que apareix cap al final, quedarà reduïda (subratllem les parts afectades) a:

```
(if (not BL*EX)
    (progn
      (if NORM-Z
          (INSPARCIAL (setq K (/ OX* OX**)) K K X*)
          (progn
            (if B (command "SCP" "Z" O X
                           "SCP" "X" 90))
            (INSPARCIAL (setq J (/ 1 OX**) K (* OXY** J))
                        (* (distance Z Z**) J) (* K (/ OX* OXY*)) XY**)
            (command "SCP" "PR" "SCP" "PR")))
      (command "SCP" "Z" O X**
               "BLOQUE" BLOC* O SS "")
      (if OO-4 (command "BLOQUE" ABLOC* O SA ""))
      (command "SCP" "PR")))
```

- Pel que fa a **C:DESCOMPOK**, cap al final hi afegirem l'assignació subratllada:

```
      (while (and (setq E NEXTE)
                  (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                  (wcmatch (setq BLOC* (cdr (assoc 2 LE)))
                           (setq A (strcat "ATRIBATIP_" (itoa K) "_DE_*"))))
             (command "DESCOMP" E ())
             (setq NEXTE (entnext E) E (ssname (ssget "P") 0)
                   E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*)))
                   E* (if (wcmatch BLOC* (strcat A " " N)) (entnext E*) E*))
             (REST-OBLIC))
```

Quan, poc abans de presentar aquest codi, ens referíem a les mancances que ens impedien actualitzar el genèrics 2D ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***> a partir del primitius 2D ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**>, estàvem perfilant de manera implícita la solució: igual que fèiem amb els **BLOC**s compostos exclusivament per atributs P i N, treballar com sempre amb el tàndem format per **BLOC-1** i **BLOC-2**, encara que en aquests casos el primer fos un bloc buit de contingut. De tota manera, hem complicat les coses just fins on era inevitable: cada bloc portador seguirà tenint un atribut atípic com a únic contingut, i només als seus genèrics els incorporarem una inserció de bloc buit (prèviament creat amb el nom BLOC_NUL) efectuada amb els mateixos paràmetres que la del portador de l'atribut; aquesta última l'explosionarem per retornar la virginitat a l'atribut i deixar-lo en condicions de suportar una nova assignació de valor en inserir el genèric on ha trobat refugi en companyia de la inserció buida. En definitiva, només complicarem

ATRIBATIP_<núm.>_DE_<nom_de_**BLOC***>, introduint-li el polissó que farà de testimoni dels paràmetres d'inserció a **REDEF-BLOC*S**, quan ens posem a actualitzar aquests genèrics 2D, sense cap necessitat de desdoblar ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**>. Han estat aquesta homologació formal i l'activació de **OO-2** des de **INSERT*** allò que ens ha permès prescindir del argument **PRAL** de **INSERT*** i recórrer a **INSPARCIAL**. La incorporació d'una inserció de bloc buit als derivats genèrics dels portadors 2D de cada atribut atípic també és causa de la discriminació introduïda al final de **C:DESCOMPOK**: si el paral·lelogram d'encaix determinat per la tríada de punts és un rectangle, inserirem directament el bloc portador (amb l'atribut atípic com a únic contingut) i, si l'anomenem **BLOC***, el nom intern **E*** de l'atribut atípic serà **(cdr (assoc -2 (tblsearch "BLOCK" BLOC*)))**; però si no ho és, caldrà que el **BLOC*** inserit sigui un genèric del portador (que contindrà una inserció de bloc buit i l'atribut atípic, en aquest ordre), raó per la qual el nom intern **E*** passarà a ser **(entnext (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))**. Tanmateix, si després de l'explosió de cada **BLOC*** (propagada a partir de l'explosió de principal) executem **(ssname (ssget "P") 0)**, sempre en resultarà el nom intern de l'atribut: si **BLOC*** era el bloc portador, perquè l'atribut atípic serà l'únic objecte del conjunt de selecció; si era el seu derivat genèric, perquè els dos objectes que componen el conjunt, inserció del portador i atribut, hauran permutat les seves posicions.

Tot i que de seguida veurem com encara no encaixen totes les peces, sí que podríem recapitular en el sentit de presentar una sinopsi dels diferents blocs que poden aparèixer en aplicar ***INS2D*/*INS3D*** a **BLOC**. Dintre de cada apartat, el presentarem en l'ordre següent: 1) blocs substituts, hi hagi o no atributs P o N; 2) blocs genèrics de primer i segon grau, hi hagi o no atributs P o N; 3) blocs portadors d'atributs atípics, genèric de segon grau amb la carcassa de punts i genèrics de primer grau dels penúltims (o bé blocs substituts per a insercions ortogonals), només si hi ha atributs atípics. I, per fer-ho més intel·ligible i fugir de la endimoniada notació híbrida que hem estat suportant, suposarem que **BLOC** s'anomena *B*, té 3 atributs atípics y usarem valors concrets:

INS3D:

1 Encaix: paral·lelepípede oblic de base romboïdal (ni **NORM-XY** ni **NORM-Z**)

B_SENSE_ATRIBS	*B_SENSE_ATRIBS_397643*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B	*ATRIBS_DE_B_397643*	*ATRIBATIP_2_DE_B*
	B_397643_859866	*ATRIBATIP_3_DE_B*
		ATRIBATIP_1_DE_B_866
		ATRIBATIP_2_DE_B_789
		ATRIBATIP_3_DE_B_999
		ATRIBSATIPS_DE_B_397643_859866

2 Encaix: paral·lelepípede oblic de base rectangular (**NORM-XY**, però no **NORM-Z**)

B_SENSE_ATRIBS	*B_SENSE_ATRIBS_397643*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B	*ATRIBS_DE_B_397643*	*ATRIBATIP_2_DE_B*
	B_397643_859	*ATRIBATIP_3_DE_B*
		ATRIBATIP_2_DE_B_772
		ATRIBATIP_3_DE_B_976
		ATRIBSATIPS_DE_B_397643_859

3 Encaix: paral·lelepípede recte de base romboïdal (no **NORM-XY**, però sí **NORM-Z**)

B_SENSE_ATRIBS	*B_866*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B		*ATRIBATIP_2_DE_B*
		ATRIBATIP_3_DE_B
		ATRIBATIP_1_DE_B_866
		ATRIBATIP_3_DE_B_278
		ATRIBSATIPS_DE_B_866

4 Encaix: paral·lelepípede ortogonal (**NORM-XY** i **NORM-Z**)

		ATRIBATIP_1_DE_B
		ATRIBATIP_2_DE_B
		ATRIBATIP_3_DE_B
		B_AMB_ATRIBSTIPS
		ATRIBSATIPS_DE_B

INS2D:

5 Encaix: paral·lelogram oblic (no **NORM-XY**, però sí **NORM-Z**)

B_SENSE_ATRIBS	*B_866*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B		*ATRIBATIP_2_DE_B*
		ATRIBATIP_3_DE_B
		ATRIBATIP_1_DE_B_866
		ATRIBATIP_3_DE_B_312
		ATRIBSATIPS_DE_B_866

6 Encaix: rectangle (**NORM-XY** i **NORM-Z**)

ATRIBATIP_1_DE_B
ATRIBATIP_2_DE_B
ATRIBATIP_3_DE_B
B_AMB_ATRIBSTIPS
ATRIBSATIPS_DE_B

Reblant el clau, assenyalarem que: **NLOC-1** i **NLOC-2** s'anomenen *B_AMB_ATRIBSTIPS* i *ATRIBSATIPS_DE_B* respectivament (casos 4 i 6); **BLOC-1** i **BLOC-2**, *B_SENSE_ATRIBS* i *ATRIBS_DE_B* (casos 1, 2, 3 i 5); els genèrics de primer grau **BLOC-1*** i **BLOC-2***, *B_SENSE_ATRIBS_397643* i *ATRIBS_DE_B_397643* (casos 1 i 2); els de segon **BLOC**** i **BLOC-2****, *B_397643_859866* i *ATRIBSATIPS_DE_B_397643_859866* (cas 1), o *B_397643_859* i *ATRIBSATIPS_DE_B_397643_859* (cas 2). Quant als genèrics de primer grau anomenats *B_866* i *ATRIBSATIPS_DE_B_866* (casos 3 i 5), en el text explicatiu no ens hi hem referit de cap manera específica, perquè tant (o tant poc) sentit tindria parlar de **BLOC*** i **BLOC-2*** (un asterisc suggereig un grau, però **BLOC-2*** ja tenia un significat) com de **BLOC**** i **BLOC-2**** (s'assimila a la notació dels casos 1 i 2, però els dos asteriscs suggereixen que són genèrics de segon grau); per acabar-ho d'adobar, queda la ja comentada ambigüitat de **BLOC*** en el codi, que en alguns casos representa el que aquí anomenem **BLOC****. En una altra línia direm que, si *B* té atributs atípics, pot ser que no tots els seus blocs portadors necessitin el concurs d'un genèric per poder encaixar en el paral·lelogram definit per la tríada de punts que els pertoca: això passarà quan el paral·lelogram esdevingui rectangle i l'encaix el pugui assegurar el propi bloc portador. Per exemple, en els casos 4 i 6 el portador de *ATRIBATIP_3_DE_B* només podrà prescindir de genèric si **NLOC-1** i **NLOC-2** queden inserits amb uns factors $\mathbf{E_x = E_y}$; altrament, sorgirà un genèric que dependrà d'aquests factors i també de $\mathbf{E_z}$ (si $\mathbf{E_y = 2\,E_x}$, serà *ATRIBATIP_3_DE_B_622* en el cas 4, però *ATRIBATIP_3_DE_B_775* en el cas 6).

Ja a la VERSIÓ 17+, la irrupció en escena del segon tàndem **NLOC-1**/**NLOC-2** ens havia dut a substituir la senzilla condició **(and BL1EX BL2EX)**, que identificava els casos exclosos de la necessitat d'una anàlisi en profunditat, per la més complexa

(or (and NORM NL1EX NL2EX (not (XOR BL1EX BL2EX)))
(and (not NORM) BL1EX BL2EX (not (XOR NL1EX NL2EX))))

Però no totes les altres situacions requereixen una inspecció en profunditat, com fèiem allà i hem seguit fent a la VERSIÓ 18+: en rigor, només la situació inicial **(not (or NL1EX NL2EX BL1EX BL2EX))**, les que responen molt probablement al propòsit de l'usuari de forçar la introducció en el sistema ***INS2D***/***INS3D*** d'un **BLOC** redefinit quan ja **NLOC-1** i **NLOC-2** i/o **BLOC-1** i **BLOC-2** havien estat creats, tot destruint **NL2EX** i/o **BL2EX** (**(and NL1EX (not NL2EX))** i/o **(and BL1EX (not BL2EX))**), i també les que poden ser resultat d'un fet fortuït (prèvia l'eliminació de les insercions, també fortuïta o no), per destrucció de **NL1EX** i/o **BL1EX** (**(and (not NL1EX) NL2EX)** i/o **(and (not BL1EX) BL2EX)**), comportaran una reconstrucció feta a partir de **BLOC**.

És clar que en el supòsit **(and (not NORM) NL1EX NL2EX (not BL1EX) (not BL2EX))** cal definir **BLOC-1** i **BLOC-2**, i que en **(and NORM (not NL1EX) (not NL2EX) BL1EX BL2EX)** cal definir **NLOC-1** i **NLOC-2**, però hi ha una qüestió insignificant en aparença que obligarà a introduir a la VERSIÓ 18+ força més canvis que els que aquesta havia representat per a la VERSIÓ 17+: en tots dos casos podria esdevenir-se que **BLOC** fos redefinit, una vegada creat el primer tàndem de substituts (**NLOC-1** i **NLOC-2** en el primer cas, o **BLOC-1** i **BLOC-2** en el segon), sense que de moment hi hagués la més mímima intenció que ***INS2D***/***INS3D*** registrés el canvi. Doncs bé: si, com havíem decidit des de la VERSIÓ 9, el millor per forçar la redefinició de **BLOC-1** i **BLOC-2** (ampliats ara a **NLOC-1** i **NLOC-2**, si s'escau) és destruir **BLOC-2** (o bé **NLOC-2**, si s'escau), perquè no depèn de cap inserció consolidada, no podem permetre que, després de les primeres insercions obliqües d'un **BLOC** amb atributs atípics (amb la creació de **BLOC-1** i **BLOC-2**, entre d'altres), una eventual redefinició de **BLOC** es manifesti en **NLOC-1** i **NLOC-2** (o que, fins i tot, aquests substituts no arribin a aparèixer, quan el canvi comporti la supressió total dels atributs atípics) sense el nostre consentiment (és a dir, sense abans esborrar **BLOC-2**), en realitzar la primera inserció ortogonal. Com tampoc no seria acceptable que, després de les primeres insercions ortogonals d'un **BLOC** amb atributs atípics (amb la consegüent creació de **NLOC-1** i **NLOC-2**, entre d'altres), una eventual redefinició de **BLOC** es manifestés a **BLOC-1** i **BLOC-2** en realitzar la primera inserció obliqua, sense el nostre consentiment (és a dir, sense abans esborrar **NLOC-2**). En realitat, aquests casos hauríen de rebre el mateix tracte que dues insercions ortogonals [obliqües] seguides, entre les quals s'hagués redefinit **BLOC** però sense executar el protocol d'acollida del nou bloc. L'única diferència és que aquí ja no n'hi haurà prou a

seguir usant **NLOC-1** i **NLOC-2** [**BLOC-1** i **BLOC-2**], dipositaris de la informació pre-redefinició, sinó que haurem de prendre-la d'aquest tàndem i reorganitzar-la per transferir-la al nou tàndem **BLOC-1** i **BLOC-2** [**NLOC-1** i **NLOC-2**] que s'ha de crear: les funcions **N->B** i **B->N** s'encarregaran d'aquesta missió, formant les tres llistes **OO-1** (atributs Constants i d'altres objectes), **OO-2** (atributs P i N típics) i **OO-3** (punts per situar els atributs P i N atípics) per realitzar el traspàs, ja que com sabeu **BLOC-1** només conté components **OO-1**, i **BLOC-2** aplega components **OO-2** i **OO-3**, mentre que **NLOC-1** aplega components **OO-1** i **OO-2**, i **NLOC-2** només conté components **OO-3**. En principi, s'havia pensat en dues funcions diferenciades, cadascuna de les quals s'aplicaria primer a **BLOC-1** [**NLOC-1**] i després a **BLOC-2** [**NLOC-2**]:

```
(defun B->N (B / C)
  (setq C (= B BLOC-1)
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E)
                   AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if C
                 (setq AT-C (if AT-C T AT)
                       OO-1 (cons LE OO-1))
                 (if AT
                     (setq OO-2 (cons LE OO-2))
                     (setq OO-3 (cons LE OO-3))))
             (setq E (entnext E)))))

(defun N->B (B / C)
  (setq C (= B NLOC-1)
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E)
                   AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if C
                 (if (and AT (= (logand (cdr (assoc 70 LE)) 2) 0))
                     (setq OO-2 (cons LE OO-2))
                     (setq AT-C (if AT-C T AT)
                           OO-1 (cons LE OO-1)))
                 (setq OO-3 (cons LE OO-3)))
             (setq E (entnext E)))))
```

Però serà fàcil unificar-les sota el nom **B->N**, només afegint un argument lògic **A** que indicarà si el nom és o no apropiat. Ens entretindrem a veure'n l'evolució a partir de la formulació més immediata, perquè si no ho féssim així podria haver-hi dificultats per interpretar la versió depurada que figurarà en el codi de conjunt:

```
(defun B->N (A B / C)
  (setq C (= B (if A BLOC-1 NLOC-1))
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E)
                   AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if C
                 (if A
                     (setq AT-C (if AT-C T AT)
                           OO-1 (cons LE OO-1))
                     (if (and AT (= (logand (cdr (assoc 70 LE)) 2) 0))
                         (setq OO-2 (cons LE OO-2))
                         (setq AT-C (if AT-C T AT)
                               OO-1 (cons LE OO-1))))
                 (if A
                     (if AT
                         (setq OO-2 (cons LE OO-2))
                         (setq OO-3 (cons LE OO-3)))
                     (setq OO-3 (cons LE OO-3))))
             (setq E (entnext E)))))
```

Però fixem-nos que l'expressió
```
  (setq AT-C (if AT-C T AT)
        OO-1 (cons LE OO-1))
```
surt en dues situacions que tenen en comú **C** i que es poden refondre en una de sola
```
  (or A (and (not A) (not (and AT (= (logand (cdr (assoc 70 LE)) 2) 0)))))
```
que podem posar en la forma
```
  (or A (not (or A (and AT (= (logand (cdr (assoc 70 LE)) 2) 0)))))
```
La complementària d'aquesta,
```
  (not (or A (not (or A (and AT (= (logand (cdr (assoc 70 LE)) 2) 0))))))
```
es pot representar també com
```
  (and (not A) (or A (and AT (= (logand (cdr (assoc 70 LE)) 2) 0))))
```
que, com que la condició necessària **(not A)** invalida la suficient **A**, equival a
```
  (and (not A) (and AT (= (logand (cdr (assoc 70 LE)) 2) 0)))
```
que reduirem a
```
  (and (not A) AT (= (logand (cdr (assoc 70 LE)) 2) 0))
```

Així doncs, si es dóna **C**,
```
  (if (and (not A) AT (= (logand (cdr (assoc 70 LE)) 2) 0))
      (setq OO-2 (cons LE OO-2))
      (setq AT-C (if AT-C T AT)
            OO-1 (cons LE OO-1)))
```

D'altra banda, l'expressió
```
  (setq OO-3 (cons LE OO-3))
```
surt en dues situacions que tenen en comú **(not C)** i que es poden refondre en
```
  (or (and A (not AT)) (not A))
```
que equival a
```
  (not (and A AT))
```

Així doncs, si es dóna **(not C)** i usem la complementària, tindrem
```
  (if (and A AT)
      (setq OO-2 (cons LE OO-2))
      (setq OO-3 (cons LE OO-3)))
```

Aplegant ara els dos estats de **C**, la funció única quedaria així:

```
(defun B->N (A B / C)
  (setq C (= B (if A BLOC-1 NLOC-1))
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E)
                   AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if C
                 (if (and (not A) AT (= (logand (cdr (assoc 70 LE)) 2) 0))
                     (setq OO-2 (cons LE OO-2))
                     (setq AT-C (if AT-C T AT)
                           OO-1 (cons LE OO-1)))
                 (if (and A AT)
                     (setq OO-2 (cons LE OO-2))
                     (setq OO-3 (cons LE OO-3))))
             (setq E (entnext E)))))
```

Però considerant que les dues situacions en què es dóna
```
  (setq OO-2 (cons LE OO-2))
```
es poden refondre en la condició
```
  (or (and C (and (not A) AT (= (logand (cdr (assoc 70 LE)) 2) 0)))
      (and (not C) (and A AT)))
```
que, prescindint d'operadors **and** sobrers, queda en
```
  (or (and C (not A) AT (= (logand (cdr (assoc 70 LE)) 2) 0))
      (and (not C) A AT))
```

Com que **AT** figura en les dues funcions **and**, podem deixar-la en
```
  (and AT (or (and C (not A) (= (logand (cdr (assoc 70 LE)) 2) 0))
              (and (not C) A))
```
i així, la funció **B->A** queda reduïda a la forma en què la veurem més endavant

```
(defun B->N (A B / C)
  (setq C (= B (if A BLOC-1 NLOC-1))
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E)
                   AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if (and AT (or (and A (not C))
                             (and (not A) C
                                  (= (logand (cdr (assoc 70 LE)) 2) 0))))
                 (setq OO-2 (cons LE OO-2))
                 (if C
                     (setq AT-C (if AT-C T AT)
                           OO-1 (cons LE OO-1))
                     (setq OO-3 (cons LE OO-3))))
             (setq E (entnext E)))))
```

Potser fóra bo tenir a la vista totes les situacions que es poden presentar, en funció de les variables booleanes **NL1EX**, **NL2EX**, **BL1EX**, **BL2EX** i **NORM**, i el tracte que mereixen, i res de més definitiu que muntar-ne allò que en electrònica digital anomenen "taula de veritat", referida a un node del circuït on podem considerar diverses entrades i una sortida. Aquí considerarem com a ínputs les 5 variables binàries esmentades, que adoptaran el valor **1** o **0** segons que siguin o no veritat en cadascun dels $2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 31$ casos possibles, i l'output, representat per la sisena columna, especificarà els tàndems **NLOC-1**/**NLOC-2** i/o **BLOC-1**/**BLOC-2** que cal crear de nou o recrear (si, en lloc de fer-ho a partir del bloc original, el construïm a partir de l'altre tàndem, anirà seguit d'asterisc). A l'esquerra, identificarem cada cas-fila amb el nombre decimal equivalent al definit en sistema binari pel conjunt de les 5 variables-columna.

cas núm.	NL1EX	NL2EX	BL1EX	BL2EX	NORM	creació/recreació substituts	
0	**0**	**0**	**0**	**0**	**0**	**-**	**BLOC-1/BLOC-2**
1	**0**	**0**	**0**	**0**	**1**	**NLOC-1/NLOC-2**	**-**
2	**0**	**0**	**0**	**1**	**0**	**-**	**BLOC-1/BLOC-2**
3	**0**	**0**	**0**	**1**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
4	**0**	**0**	**1**	**0**	**0**	**-**	**BLOC-1/BLOC-2**
5	**0**	**0**	**1**	**0**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
6	**0**	**0**	**1**	**1**	**0**	**-**	**-**
7	**0**	**0**	**1**	**1**	**1**	**NLOC-1/NLOC-2***	**-**
8	**0**	**1**	**0**	**0**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
9	**0**	**1**	**0**	**0**	**1**	**NLOC-1/NLOC-2**	**-**
10	**0**	**1**	**0**	**1**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
11	**0**	**1**	**0**	**1**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
12	**0**	**1**	**1**	**0**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
13	**0**	**1**	**1**	**0**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
14	**0**	**1**	**1**	**1**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
15	**0**	**1**	**1**	**1**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
16	**1**	**0**	**0**	**0**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
17	**1**	**0**	**0**	**0**	**1**	**NLOC-1/NLOC-2**	**-**
18	**1**	**0**	**0**	**1**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
19	**1**	**0**	**0**	**1**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
20	**1**	**0**	**1**	**0**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
21	**1**	**0**	**1**	**0**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
22	**1**	**0**	**1**	**1**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
23	**1**	**0**	**1**	**1**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
24	**1**	**1**	**0**	**0**	**0**	**-**	**BLOC-1/BLOC-2***
25	**1**	**1**	**0**	**0**	**1**	**-**	**-**
26	**1**	**1**	**0**	**1**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
27	**1**	**1**	**0**	**1**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
28	**1**	**1**	**1**	**0**	**0**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
29	**1**	**1**	**1**	**0**	**1**	**NLOC-1/NLOC-2**	**BLOC-1/BLOC-2**
30	**1**	**1**	**1**	**1**	**0**	**-**	**-**
31	**1**	**1**	**1**	**1**	**1**	**-**	**-**

Això ens permetrà esquematitzar el flux principal de **SEGR-ATRIBS** amb referència a l'identificador numèric de cada cas:

```
(if (or (and NORM NL1EX NL2EX (not (XOR BL1EX BL2EX)))
        (and (not NORM) BL1EX BL2EX (not (XOR NL1EX NL2EX))))
    (casos 6, 25, 30 i 31: no cal crear res)
    (if (and (not NL1EX) (not NL2EX) BL1EX BL2EX)
        (cas 7: creació de NLOC-1 i NLOC-2 des de BLOC-1 i BLOC-2)
        (if (and NL1EX NL2EX (not BL1EX) (not BL2EX))
            (cas 24: creació de BLOC-1 i BLOC-2 des de NLOC-1 i NLOC-2)
            (progn
              (if (or NL1EX NL2EX NORM)
                  (casos 1, 3, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16,
                         17, 18, 19, 20, 21, 22, 23, 26, 27, 28 i 29:
                         creació de NLOC-1 i NLOC-2 des de BLOC))
              (if (or BL1EX BL2EX (not NORM))
                  (casos 0, 2, 3, 4, 5, 8, 10, 11, 12, 13, 14, 15,
                         16, 18, 19, 20, 21, 22, 23, 26, 27, 28 i 29:
                         creació de BLOC-1 i BLOC-2 des de BLOC)))))
```

Com que a **SEGR-ATRIBS** i més enllà d'aquesta funció encara hi ha temes puntuals que poden quedar foscos, abans de passar sense solució de continuïtat de l'estratègia esbossada al codi, aclarirem algunes qüestions com hem fet abans amb **B->N:**

- Ara ja podem assegurar que la decisió adoptada a VERSIÓ 17+ sobre l'escapçada de vèrtexs era correcta, pel que fa a l'eliminació de **BLOC-2** i/o **NLOC-2** com a mitjà més efectiu per desencadenar la reconstrucció completa dels blocs que, dins de **INSERTOK**, es reparteixen la informació continguda a **BLOC:** la correspondència entre ells estarà assegurada pel propi sistema, perquè només els casos **7** i **24** de la taula presentada poden desembocar en aquesta situació, i ja hem vist que la creació del nou tàndem de substituts (**NLOC-1**/**NLOC-2** en **7** i **BLOC-1**/**BLOC-2** en **24**) no es fonamentava en **BLOC** sinò en l'altre tàndem.

- Ens fem enrera de la decisió d'eliminar amb ***LIMPIA*** els blocs auxiliars **NLOC-2** i **BLOC-2** perquè d'una banda evitàvem l'aparició del missatge *Redefiniendo el bloque "ATRIBS..."*, però de l'altra ens asseguràvem de veure el propi d'aquesta ordre, *Suprimiendo bloque "ATRIBS..."*.

- També ens farem enrera de la destrucció dels genèrics **BLOC-2***, perquè l'únic que en traiem és l'aparició de missatges *Suprimiendo bloque "ATRIBS_DE_..."*. Potser si **REDEF-BLOC*S** hagués d'afrontar l'eventualitat d'una redefinició de **BLOC-2*** substituint **command** per una llista sense avaluar **(eval (append '(command ...)))**, per integrar-hi l'argument addicional **"S"** (una resposta afirmativa al missatge *El bloque ... ya existe. ¿Desea volver a definirlo? [Sí/No] <N>:*), la mesura tindría algun sentit: poder treballar directament amb **command**, incloent-hi **"S"** després de **BLOC-1*** però no després de **BLOC-2***. Tanmateix, ja hem aclarit en el capítol precedent que tot plegat ho resoldríem fent ***EXPERT*** = **2**. A més, únicament evitarem perdre el temps a **REDEF-BLOC*S**, actualitzant un genèric de primer grau obsolet, si prèviament hem esborrat les insercions dels derivats **BLOC**** de segon grau i hem destruït també el company de tàndem **BLOC-1***: si no es dóna la primera condició, ***LIMPIA*** no podrà actuar; si no es dóna la segona, **REDEF-BLOC*S** tornarà a crear **BLOC-2***. I, pel que fa a **BLOC-2****, tres quarts del mateix: si **BLOC**** existeix i la llista **OO-4** no és nul·la, **REDEF-BLOC*S** tornarà a donar vida a l'esquelet de punts i l'únic guany haurà estat la irrupció de l'eco de ***LIMPIA***.

- Finalment, i en relació als blocs 2D portadors d'atributs atípics i als seus derivats, també optarem per retirar ***LIMPIA*** de l'escena: el guirigall organitzat pels missatges *Suprimiendo bloque "ATRIBATIP_..."*, *No se han encontrado entradas con el nombre "ATRIBATIP_..." o No se ha encontrado ningún bloque sin referencia* llançats des d'aquesta ordre no és pas preferible al de **entmake** *Redefiniendo el bloque "ATRIBATIP_..."*. A més, ja és pressuposar massa donar per obsolets uns blocs pel simple fet que no tenen presència en el dibuix (insercions efectives o en la definició d'un altre bloc): més val que l'usuari decideixi de quins blocs vol prescindir.

- Quan abans hem posat en evidència la necessitat de singularitzar els casos que ara anomenem **7** i **24**, evitant que els nous tàndems substituts (**NLOC-1** i **NLOC-2** en el **7**; **BLOC-1** i **BLOC-2** en el **24**) fossin creats a partir d'un **BLOC** que tal vegada havia estat redefinit, i forçant-los a treure la informació dels ja existents (**NLOC-1** i **NLOC-2** en el **7**; **BLOC-1** i **BLOC-2** en el **24**), hauríem d'haver remarcat

una asimetria que semblarà paradoxal però que hem d'assumir per ser conseqüents amb tot el discurs que ens ha dut fins aquí. Una primera inserció ***INS2D***/***INS3D*** sempre donarà lloc a **BLOC-1** i **BLOC-2** si és obliqua (**0**), però si és ortogonal (**1**) sols generarà **NLOC-1** i **NLOC-2** quan **BLOC** contingui atributs atípics; si no en té, la inserció treballarà directament amb **BLOC**. Però dels casos **7** i **24** sempre en sortirà l'altre tàndem i, si això no té res d'estrany en el cas **24** (necessitem **BLOC-1** i **BLOC-2** tant si **BLOC** té atributs atípics com si no), en el **7** no resulta tan evident (ja hem dit que creàvem uns **NLOC-1** i **NLOC-2** basats en la informació continguda a **BLOC-1** i **BLOC-2** perquè, si la treiéssim de **BLOC**, podria passar que aquest hagués estat redefinit des de l'última inserció ***INS2D***/***INS3D*** registrada, i que acabéssim inserint-lo actualitzat sense haver seguit el protocol establert). Tot això es traduirà en un comportament que no és arbitrari però que ho sembla, en la mesura que la resposta depèn de circumstàncies accidentals, alienes a la composició de **BLOC** i a les condicions geomètriques de la seva inserció: si **BLOC** no té atributs atípics (o, més encara, si no té atributs P ni N), mentre no hi hagi insercions obliqües cap **NLOC-1** ni **NLOC-2** no irrompran en escena per moltes insercions ***INS2D***/***INS3D*** ortogonals que realitzem, perquè el bloc inserit serà el propi **BLOC** (**1**); però així que se'n produeixi una i **BLOC-1** i **BLOC-2** facin acte de presència, la primera inserció ***INS2D***/***INS3D*** ortogonal (**7**) i les subsegüents (**31**) treballaran amb **NLOC-1** (de contingut idèntic a **BLOC**) i **NLOC-2** (sense contingut), però tan aviat com es produeixi l'eliminació d'un d'aquests arxius auxiliars, de dos (que no siguin membres d'un mateix tàndem, perquè si no tornaríem al cas **7**, que reproduiria **NLOC-1** i **NLOC-2**, o al **25**, que ho deixaria tot igual), de tres o de tots quatre substituts (prèvia la destrucció de les insercions que calgui), les insercions ***INS2D***/***INS3D*** ortogonals tornaran a dependre exclusivament de **BLOC** (**1, 3, 5, 9, 11, 13, 15, 17, 19, 21, 23, 27** i **29**). De fet, en comptes de crear indiscriminadament un **NLOC-1** clavat a **BLOC** i un **NLOC-2** buit en el cas **7**, podríem estalviar-nos-ho quan el contingut conjunt de **BLOC-1** i **BLOC-2** fos equivalent al de **BLOC**, però això comportaria examinar el contingut dels tres blocs (examen que hauria de repetir-se en cada inserció ortogonal ***INS2D***/***INS3D***, tret que alguna variable global en la sessió de dibuix ens anés informant del resultat de la verificació) i tampoc no ens salvaria d'implementar en el programa les nombroses modificacions a què abans hem fet esment: que es produís en un nombre menor de casos (quan la verificació fos negativa) encara justificaria menys complicar el codi diversificant estratègies.

- Una de les coses que una decisió en aparença tan innòcua ens obliga a revisar és l'accés a la funció **VAL-ATRIBS**, que en la VERSIÓ 17+ havíem convertit en previ a l'execució de **SEGR-ATRIBS**: si resulta que, anàlogament als casos **6, 25, 30** i **31**, en els **7** i **24** la informació no s'ha de treure de **BLOC** sinó dels suplents **BLOC-1**/**BLOC-2** (**7**) o **NLOC-1**/**NLOC-2** (**24**), caldrà que la construcció de les llistes **WWAA-1** i **WWAA-2** de valors d'atribut tingui lloc un cop coneguem quina és la situació, ja no sols per l'eventual desactivació de la característica *Verificable* d'algun atribut, sinó per la reordenació que la presència d'atributs atípics hagi pogut produir en distribuir aquests components entre els blocs **NLOC-1** i/o **BLOC-2** d'una banda i els portadors ATRIBATIP_<núm.>_DE_<nom_de_**BLOC**> de l'altra: seria un desgavell absolut que ***INS2D***/***INS3D*** ens demanés els valors d'assignació en l'ordre original quan haguéssim de recórrer a **BLOC** i en un de diferent quan depenguéssim dels blocs esmentats. Com que de seguida oferirem el codi sencer de ***INS2D***/***INS3D***, ja no ens molestem a presentar separadament el de les funcions implicades, i ens limitarem a enumerar-les: **VAL-ATRIBS-1** sols processarà atributs, desproveïts ja de la característica *Verificable* (casos **6, 7, 24, 25, 30** i **31**, sobre una llista **OORD** d'atributs P i N restituïts a l'ordre inicial si n'hi ha d'atípics, i sobre la llista **OO-2** si no n'hi ha), mentre que **VAL-ATRIBS-2** (hereva de **VAL-ATRIBS**) se les veurà amb els components originals de **BLOC** (tots els altres casos) i haurà de repetir la sol·licitud de valor en els atributs amb aquesta característica; aquesta contradicció ens obligarà a prescindir d'una de les simplificacions adoptades al principi del capítol, però com a qüestió menor la deixarem per a després, per no acumular massa canvis en el trànsit d'una versió a la següent.

- L'altra és el muntatge del dispositiu que, des de **SEGR-ATRIBS**, ens permet anotar l'ordre dels atributs típics en el text **E1** i dels atípics en **E2**, i de la funció **WWAA->BLOC** que crea amb aquests textos sengles atributs *Constants* i *Invisibles* (de nom "WWAA-1" i "WWAA-2", respectivament) i els incorpora a la llista **OO-1** per tal que aquesta informació sigui emmagatzemada en **BLOC-1** o **NLOC-1** (tot això si analitzem directament **BLOC**, és a dir, fora dels casos **6, 7, 24, 25, 30** i **31**). En recuperar **BLOC-1** o **NLOC-1** amb una nova inserció ***INS2D***/***INS3D*** (que respondrà a algun dels casos esmentats, tret que haguéssim destruït arxius auxiliars), la

funció **BLOC->WWAA** s'encarregarà de capturar-la, de reordenar la lista resultant d'afegir la llista d'atributs atípics a la de típics i de passar-la a la funció **VAL-ATRIBS-1** perquè els atributs d'una i altra categoria surtin a la palestra en l'ordre **OORD** original i les respostes **WWAA** es reorganitzin de nou en funció del seu estàtus, distribuint-se entre les llistes **WWAA-1** i **WWAA-2.**

No podíem passar per alt aquest dispositiu (les llistes **E1** i **E2**, que constitueixen el vincle entre **WWAA-1** i **WWAA-2**, respectivament, i **WWAA**; la funció **WWAA->BLOC**, que les emmagatzema a **BLOC-1**, i la recíproca **BLOC->WWAA** que les captura i reutilitza), perquè és la baula perduda entre la VERSIÓ 16+, només apta per a atributa típics i última que contemplava totes les insercions obliqües ***INS2D***/***INS3D*** (no oblideu que, amb atributs atípics, la VERSIÓ 17+ només servia per a les insercions inaugurals o reinaugurals de **BLOC**) i la que ara presentem, apta per a atributs de tota mena en insercions obliqües (i ortogonals, quan n'hi hagi d'atípics):

```
; VERSIÓ 18+

(defun C:INS2D () (INSERTOK T))

(defun C:INS3D () (INSERTOK ()))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun U*V (U V)
  (list (- (* (cadr U) (last V)) (* (last U) (cadr V)))
        (- (* (last U) (car  V)) (* (car  U) (last V)))
        (- (* (car  U) (cadr V)) (* (cadr U) (car  V)))))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS-1 (OO E1E2 / I N M VD V)
  (foreach O (reverse OO)
    (setq ICVP (cdr (assoc 70 O)))
    (if (and E1E2 (or (not O) (= (logand ICVP 8) 8)))
        (setq WWAA (append WWAA '(())))
        (if (= (logand ICVP 8) 0)
            (progn
              (if (and (not I) ATREQ)
                  (progn
                    (setq I T)
                    (prompt "\nIndique valores de atributo")))
```

```
              (VATR () (cdr (assoc 2 O)) (cdr (assoc 3 O)) (cdr (assoc 1 O)))
              (setq WWAA (append WWAA (list V)))))))))

(defun VAL-ATRIBS-2 (/ N M VD V LLAA)
  (setq E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
  (while E
         (if (and (= (cdr (assoc 0 (setq LE (entget E)))) "ATTDEF")
                  (= (logand (setq ICVP (cdr (assoc 70 LE))) 10) 0))
             (progn
               (if (and (not LLAA) ATREQ)
                   (prompt "\nIndique valores de atributo"))
               (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LE))
                     (cdr (assoc 3 LE)) (cdr (assoc 1 LE)))
               (setq LLAA (cons (list W N M V) LLAA))))
         (setq E (entnext E)))
  (setq W ())
  (foreach LA LLAA
    (if (car LA)
        (progn
          (if (and (not W) ATREQ) (prompt "\nVerificar valores de atributo"))
          (VATR (cons "" W) (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq WWAA (cons V WWAA))))

(defun MAKEBLOC (BLOC/2 ATRIBS OO)
  (entmake (list '(0 . "BLOCK") (cons 2 BLOC/2) '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun MNLOC ()
  (MAKEBLOC (setq BLOC NLOC-1) (or AT-C OO-2) (append OO-2 OO-1))
  (MAKEBLOC NLOC-2 () OO-3))

(defun MBLOC ()
  (MAKEBLOC BLOC-1 AT-C OO-1)
  (MAKEBLOC BLOC-2 OO-2 (append OO-3 OO-2)))

(defun LINIA-BASE ()
  (polar (cdr (assoc 11 LE))
         (- (cdr (assoc 50 LE)) PI/2)
         (* (cdr (assoc 40 LE))
            (if (= C-74 1)
                (/ 1.0 -3)
                (if (= C-74 2) 0.5 1)))))

(defun PUNT (P) (list '(0 . "POINT") (cons 10 (trans P VZ 0)) (cons 210 VZ)))

(defun REST-OBLIC ()
  (setq A (cos (cdr (assoc 51 (entget E*)))))
  (if (not (equal A 1 Q0))
      (progn
        (setq LE (entget E)
              LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A)) (assoc 40 LE) LE)
              LE (subst (cons 41 (* (cdr (assoc 41 LE)) A)) (assoc 41 LE) LE))
        (entmod LE))))

 (defun INSPARCIAL (EX EY EZ ANG / COMPROBLIC)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast))
        COMPROBLIC (not (= EX EY EZ)))
  (if (or OO-2 OO-4)
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast))
        (setq SA (ssget "P") K -1)
```

```
        (while (setq E (ssname SA (setq K (1+ K))))
               (ssadd E SS))))
  (if (and OO-4 (or NORM-Z COMPROBLIC))
      (progn
        (setq SA (ssadd) K (if NORM-Z (- (sslength SS) (* 3 OO-4)) 1))
        (repeat (* 3 OO-4)
                (setq E (ssname SS K))
                (ssadd E SA)
                (ssdel E SS))))
  (if COMPROBLIC
      (progn
        (setq K 0 E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC-2))))
        (while (setq K (1+ K) E (ssname SS K))
               (REST-OBLIC)
               (setq E* (entnext E*))))))

(defun XOR (A B) (and (not (and A B)) (or A B)))

(defun B->N (A B / C)
  (setq C (= B (if A BLOC-1 NLOC-1))
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E)
                   AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if (and AT (or (and A (not C))
                             (and (not A) C
                                  (= (logand (cdr (assoc 70 LE)) 2) 0))))
                 (setq OO-2 (cons LE OO-2))
                 (if C
                     (setq AT-C (if AT-C T AT)
                           OO-1 (cons LE OO-1))
                     (setq OO-3 (cons LE OO-3))))
             (setq E (entnext E)))))

(defun ATRIB-ATIPIC (I)
  (tblsearch "BLOCK"
             (strcat "ATRIBATIP_" (itoa (set I (1+ (eval I)))) "_DE_" BLOC)))

(defun BLOC->WWAA (/ EE OO EEOO EO OORD N)
  (setq OO (reverse OO-1)
        E1 (car OO) E2 (cadr OO)
        E1 (read (strcat "(" (cdr (assoc 1 E1)) ")"))
        E2 (read (strcat "(" (cdr (assoc 1 E2)) ")")) N -1
        EE (repeat K
             (setq EE (cons (nth (setq N (1+ N)) E2) EE)))
        E2 (reverse EE)
        EE (append E1 E2) OO () N 0
        OO (repeat K
             (setq OO (cons (entget (cdr (assoc -2 (ATRIB-ATIPIC 'N)))) OO)))
        OO (reverse (append OO OO-2))
        EEOO (mapcar 'cons EE OO) N 0)
  (repeat (+ (length OO-2) J)
          (setq N (1+ N)
                EO (cdr (assoc N EEOO))
                OORD (cons EO OORD)))
  (VAL-ATRIBS-1 OORD T)
  (foreach E E1 (setq WWAA-1 (cons (nth (1- E) WWAA) WWAA-1)))
  (foreach E E2 (setq WWAA-2 (cons (nth (1- E) WWAA) WWAA-2)))
  (setq WWAA-1 (reverse WWAA-1)
        WWAA-2 (reverse WWAA-2)))

(defun WWAA->BLOC (/ O1 O2)
  (setq O1 (last OO-1))
  (if (and (= (cdr (assoc 0 O1)) "ATTDEF")
           (= (cdr (assoc 2 O1)) "WWAA-1")
           (= (cdr (assoc 70 O1)) 3))
      (setq OO-1 (reverse (cdr (cdr (reverse OO-1)))))))
```

```
  (if OO-4
      (setq O1 (list '(0 . "ATTDEF") '(8 . "0") '(70 . 3) '(2 . "WWAA-1")
                     '(3 . "") (cons 1 E1) '(72 . 0) '(74 . 0) '(10 0 0 0)
                     '(210 0 0 1) '(7 . "Standard") '(40 . 1) '(41 . 1))
            O2 (subst (cons 1 E2) (cons 1 E1)
                      (subst '(2 . "WWAA-2") '(2 . "WWAA-1") O1))
            OO-1 (append OO-1 (list O2 O1)))))

(defun REDEF-BLOC*S (/ LB NZ)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (if BL1EX
      (progn
        (setq J (1+ (strlen BLOC-1)) NZ NORM-Z NORM-Z ())
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC-1* (cdr (assoc 2 LB)))
                            (strcat BLOC-1 M (substr M 2)))
                   (progn
                     (setq E (cdr (assoc -2 LB))
                           LE (entget E)
                           K (cdr (assoc 41 LE))
                           BLOC-2* (strcat BLOC-2 (substr BLOC-1* J)))
                     (command "SCP" "EZ" O (cdr (assoc 210 LE)))
                     (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (command "SCP" "PR")
                     (PRO-DESHACER-I/F))))
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat N "*"))
                   (progn
                     (setq E (cdr (assoc -2 LB))
                           LE (entget E)
                           BLOC-1 (cdr (assoc 2 LE))
                           E (substr BLOC-1 J) NORM-Z (= E "")
                           BLOC-2 (strcat BLOC-2* E))
                     (if (not NORM-Z) (command "SCP" "EZ" O (cdr (assoc 210 LE))))
                     (INSPARCIAL (cdr (assoc 41 LE))
                                 (cdr (assoc 42 LE))
                                 (cdr (assoc 43 LE))
                                 (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (if (not NORM-Z) (command "SCP" "PR"))
                     (command "BLOQUE" BLOC* O SS "")
                     (if OO-4 (command "BLOQUE" (strcat "ATRIBSATIPS_DE_" BLOC*)
                                                O SA "")))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* NORM-Z NZ)))
  (if OO-4
      (progn
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2 OO-3 OO-2 OO-2 T NZ OO-4 OO-4 () J 0)
        (repeat NZ
                (setq J (1+ J) BLOC-2 (strcat "ATRIBATIP_" (itoa J) "_DE_" BL))
                (tblnext "BLOCK" T)
                (while (setq LB (tblnext "BLOCK"))
                       (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                                    (strcat BLOC-2 M))
                           (progn
                             (setq BLOC-1 "BLOC_NUL"
                                   E (cdr (assoc -2 LB))
                                   LE (entget E)
                                   K (cdr (assoc 41 LE)))
                             (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                             (command "BLOQUE" BLOC* O SS "")))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* OO-2 OO-3 OO-4 NZ)))
  (setvar "CLAYER" CAPA))
```

```
(defun J-PUNTS (LB)
  (setq E (cdr (assoc -2 LB)) J 0)
  (while E (setq J (if (= (cdr (assoc 0 (entget E))) "POINT") (1+ J) J)
                 E (entnext E)))
  J)

(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 NL1EX NL2EX BL1EX BL2EX AT AT-C NE E1 E2
                      C-10 C-11 C-72 C-74 OO-3 OZ VZ)
  (setq NORM (and NORM-XY NORM-Z)
        NLOC-1 (strcat BLOC "_AMB_ATRIBSTIPS")
        NL1EX (tblsearch "BLOCK" NLOC-1)
        NLOC-2 (strcat "ATRIBSATIPS_DE_" BLOC)
        NL2EX (tblsearch "BLOCK" NLOC-2)
        BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2) K 0
        OO-4 (while (ATRIB-ATIPIC 'K) K)
        K (if NL2EX
              (J-PUNTS NL2EX)
              (if BL2EX (J-PUNTS BL2EX)))
        K (if (> K 0) (/ K 3)) J OO-4)
  (if (< K OO-4)
      (setq OO-4 K)
      (if (> K OO-4)
          (progn
            (setq NL2EX () BL2EX ())
            (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))))
  (if (and (or NL2EX BL2EX) (not (XOR NL1EX NL2EX)) (not (XOR BL1EX BL2EX)))
      (progn
        (if BL1EX
            (progn
              (B->N T BLOC-1)
              (B->N T BLOC-2))
            (progn
              (B->N () NLOC-1)
              (B->N () NLOC-2)))
        (if OO-3
            (BLOC->WWAA)
            (if OO-2
                (progn
                  (VAL-ATRIBS-1 OO-2 ())
                  (setq WWAA-1 WWAA))))
        (if (and (not NL1EX) NORM)
            (MNLOC)
            (if (not (or BL1EX NORM)) (MBLOC))))
      (progn
        (if (= (logand (cdr (assoc 70 (tblsearch "BLOCK" BLOC))) 2) 2)
            (VAL-ATRIBS-2))
        (setq OO-4 () NE 0 E1 "" E2 ""
              E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (setq ICVP (cdr (assoc 70 LE))) 2) 0))
                   (progn
                     (setq NE (1+ NE)  C-72 (cdr (assoc 72 LE))
                           C-74 (cdr (assoc 74 LE))
                           LE (if (= (logand ICVP 4) 4)
                                  (subst (cons 70 (- ICVP 4)) (assoc 70 LE) LE)
                                  LE)
                           LE (if (= C-72 4)
                                  (subst (cons 72 (setq C-72 1))
                                         (cons 72 4)
                                         (subst (cons 74 (setq C-74 2))
                                                (assoc 74 LE)
                                                LE))
                                  LE)
```

```
                            LE (if (and (< C-72 3) (> C-74 0))
                                   (subst (cons 74 0)
                                          (assoc 74 LE)
;;                                        (subst (cons 11 (LINIA-BASE))
;;                                               (assoc 11 LE)
;;                                               LE))
;;   Podeu usar les 3 línies precedents com alternativa a les 5 línies subsegüents
                                          (if (> C-72 0)                          ;;
                                              (subst (cons 11 (LINIA-BASE))       ;;
                                                     (assoc 11 LE)                ;;
                                                     LE)                          ;;
                                          LE))                                    ;;
                                   LE)
                            OZ (last (assoc 10 LE))
                            VZ (cdr (assoc 210 LE)))
                      (if (and (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))
                          (progn
                            (if (= (logand (cdr (assoc 70 LE)) 10) 0)
                                (setq WWAA-1 (append WWAA-1 (list (car WWAA)))
                                      WWAA (cdr WWAA)))
                            (setq E1 (strcat E1 " " (itoa NE))
                                  OO-2 (cons LE OO-2)))
                          (progn
                            (if (= (logand (cdr (assoc 70 LE)) 10) 0)
                                (setq WWAA-2 (append WWAA-2 (list (car WWAA)))
                                      WWAA (cdr WWAA)))
                            (setq E2 (strcat E2 " " (itoa NE))
                                  OO-3 (cons (PUNT (list 0 0 OZ)) OO-3)
                                  OO-3 (cons (PUNT (list 1 0 OZ)) OO-3)
                                  OO-3 (cons (PUNT (list 0 1 OZ)) OO-3)
                                  C-10 (assoc 10 LE)
                                  C-11 (assoc 11 LE)
                                  OO-4 (cons (subst (list 10 (cadr C-10)
                                                          (caddr C-10) 0)
                                                    C-10
                                                    (subst (list 11 (cadr C-11)
                                                                 (caddr C-11) 0)
                                                           C-11
                                                           (subst '(210 0 0 1)
                                                                  (assoc 210 LE)
                                                                  LE)))
                                             OO-4)))))
                       (setq OO-1 (cons LE OO-1)
                             AT-C (if AT-C T AT)))
                 (setq E (entnext E)))
           (if (or OO-4 (not NORM))
               (progn
                 (command "REGENT"
                          "CECOLOR" "PORCAPA"
                          "CELTYPE" "PORCAPA"
                          "CELWEIGHT" -1)
                 (WWAA->BLOC)
                 (if (or NL1EX NL2EX NORM) (MNLOC))
                 (if (or BL1EX BL2EX (not NORM)) (MBLOC))
                 (if OO-4
                     (progn
                       (if (not (tblsearch "BLOCK" "BLOC_NUL"))
                           (MAKEBLOC "BLOC_NUL" () ()))
                       (setq J 0)
                       (foreach O (reverse OO-4)
                                (MAKEBLOC (strcat "ATRIBATIP_" (itoa (setq J (1+ J)))
                                                  "_DE_" BL) T (list O)))
                       (setq OO-4 (length OO-4))))
                 (if (or BL1EX OO-4) (REDEF-BLOC*S))
                 (setvar "CECOLOR" COL)
                 (setvar "CELTYPE" TLIN)
                 (setvar "CELWEIGHT" GLIN))))))
```

```
(defun CALCULA (U V OV AGUT U* U** OU* OU** V**)
  (setq A (polar O (- (angle O V) PI/2) OV)
        W (if AGUT (angle A U) (angle U A))
        B (mapcar '/ (mapcar '+ A U) '(2 2 2))
        W (angle O (polar B W (distance O B))))
  (set U* (inters U (polar U W 1) O B ()))
  (set U** (inters O (polar O (+ W PI/2) 1) U (eval U*) ()))
  (set OU* (distance O (eval U*)))
  (set OU** (distance O (eval U**)))
  (set V** (inters V (polar V W 1) O (eval U**) ())))

(defun FIX+ (O / N)
  (setq N (itoa (if (> (- O (setq N (fix O))) 0.5) (1+ N) N)))
  (repeat (- (strlen M) 1 (strlen N)) (setq N (strcat "0" N)))
  N)

 (defun INSERT** (BLOC X Y Z G / JJ KK SSAA)
  (command "INSERT" (strcat "ATRIBSATIPS_DE_" BLOC) O "XYZ" X Y Z G
           "DESCOMP" (entlast)
           "SCP" "")
  (setq SSAA (ssget "P") JJ 0 KK -1)
  (while (setq JJ (1+ JJ) BLOC (strcat "ATRIBATIP_" (itoa JJ) "_DE_" BL)
               LE (tblsearch "BLOCK" BLOC))
         (setq LE (entget (cdr (assoc -2 LE)))
               WA (if (= (logand (cdr (assoc 70 LE)) 10) 0) (car WWAA-2))
               WWAA-2 (if WA (cdr WWAA-2))
               KK (1+ KK) O (cdr (assoc 10 (entget (ssname SSAA KK))))
               KK (1+ KK) X (cdr (assoc 10 (entget (ssname SSAA KK))))
               KK (1+ KK) Y (cdr (assoc 10 (entget (ssname SSAA KK)))))
         (command "SCP" "3" O X Y)
         (setq OX (distance O X) OY (distance O Y) OZ OX
               O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (U*V X Y)
               Z-XY O I (if (< (last Z) 0) -1 1)
               BLOC-1 "BLOC_NUL" BLOC-2 BLOC
               NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                              (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
               NORM-Z T OO-2 T OO-4 ()
               WWAA-1 (if WA (list WA)))
         (if NORM-XY
             (eval (append '(command "INSERT" BLOC O OX OY 0)
                           (if ATREQ WWAA-1)))
             (INSERT*))
         (command "SCP" "PR"))
  (command "SCP" "PR"
           "BORRA" SSAA ""))

(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** Z-XY* XY XY* XY** OXY* OXY** SA SS
                  ABLOC*)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X Y* (polar O PI/2 OX) Y** O OX* OX OX** OX
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'OX* 'OX** 'Y**)
        (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
              A (inters Z-XY (polar Z-XY W 1) O X** ())
              Z-XY* (polar A (angle A Z-XY)
                           (/ (* (distance A Z-XY) (distance X* X**))
                              (distance X X**)))))))
  (if NORM-Z
      (setq BLOC* (strcat BLOC "_"))
      (progn
        (setq X (inters X* Y* O Z-XY* ())
              X (if X X Z-XY*)
              Z-XY (- (angle O X) (angle Y* X*))
```

```
            Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
            Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
            Z (trans (list (car Z-XY*) (cadr Z-XY*) (last Z)) 1 0))
       (command "SCP" "Z" O X
                "SCP" "X" (* I 90))
       (setq XY (list OX* 0 0)
            Z (trans Z 0 1))
       (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'OXY* 'OXY** 'Z**)
       (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                           (FIX+ (/ OXY** OXY* Q0)))
            BLOC* (strcat BLOC K)
            BLOC-1* (strcat BLOC-1 K)
            BLOC-2* (strcat BLOC-2 K)
            BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
       (command "SCP" "PR" "SCP" "PR")
       (if (not BL*EX)
           (progn
             (INSPARCIAL (setq K (/ OXY* OXY**)) K K X*)
             (command "SCP" "Z" O X
                      "SCP" "X" 90
                      "GIRA" SS "" O XY*
                      "SCP" "Z" O XY**
                      "BLOQUE" BLOC-1* O (setq E (ssname SS 0)) ""
                      "BLOQUE" BLOC-2* O (ssdel E SS) ""
                      "SCP" "PR")))
       (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*
            BLOC* (strcat BLOC* "_" (FIX+ (/ ZZ** OXY** Q0))))))
  (if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
  (if OO-4 (setq ABLOC* (strcat "ATRIBSATIPS_DE_" BLOC*)))
  (if (or NORM-Z BL*EX)
 (setq BL*EX (and (tblsearch "BLOCK" BLOC*)
                  (if OO-4 (tblsearch "BLOCK" ABLOC*) T))))
  (if (not BL*EX)
      (progn
        (if NORM-Z
            (INSPARCIAL (setq K (/ OX* OX**)) K K X*)
            (progn
              (if B (command "SCP" "Z" O X
                             "SCP" "X" 90))
              (INSPARCIAL (setq J (/ 1 OX**) K (* OXY** J))
                          (* (distance Z Z**) J) (* K (/ OX* OXY*)) XY**)
              (command "SCP" "PR" "SCP" "PR")))
        (command "SCP" "Z" O X**
                 "BLOQUE" BLOC* O SS "")
        (if OO-4 (command "BLOQUE" ABLOC* O SA ""))
        (command "SCP" "PR")))
  (command "CELWEIGHT" GLIN
           "CELTYPE" TLIN
           "CECOLOR" COL
           "CLAYER" CAPA)
  (eval (append '(command "INSERT" BLOC* O "XYZ" OX** (setq J (distance Y Y**))
                          (setq K (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1)
                                     OX** I)) X**)
                (if ATREQ WWAA-1)))
  (if OO-4 (INSERT** BLOC* OX** J K X**)))

(defun INSERTOK (2D / Q0 2*PI PI/2 M N CAPA COL TLIN GLIN ECO CTRL--ECO OSN ATDIA
                      ATREQ EXPERT BL BLOC BLOC* BLOC-1 BLOC-2 BLOC-1* BLOC-2*
                      BL*EX ICVP WWAA WWAA-1 WWAA-2 WA O X Y Z UV Z-XY OX OY OZ
                      X-Y X-O-Y A B E LE E* I J K W NORM-XY NORM-Z OO-1 OO-2 OO-4)
  (setq Q0 0.001 M "_")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq M (strcat M "#")))
  (setq 2*PI (* PI 2)
        PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
```

```
      ECO (getvar "CMDECHO")
      OSN (getvar "OSMODE")
      ATDIA (getvar "ATTDIA")
      ATREQ (= (getvar "ATTREQ") 1)
      EXPERT (getvar "EXPERT")
      O (getvar "INSNAME")
      BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
      BLOC (if (= BLOC "")
               (if (> O "") O (prompt "\nNombre de bloque no válido."))
               (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                            (< (cdr (assoc 70 O)) 4))
                       (and (not O) (findfile (strcat BLOC ".dwg"))))
                   BLOC (alert (if O (REFX) (RUTES)))))
      BLOC (if BLOC (strcase BLOC) (exit))
      N (strcat BLOC M) W T)
  (while W
    (setq O (getpoint "\nPrecise punto de inserción: "))
    (foreach P (if 2D '("X" "Y") '("X" "Y" "Z"))
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                  " desde el punto de inserción: "))
            W (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC  " <1>: "))
            W (if W W 1))
      (set (read P) (mapcar '(lambda (CO CA) (+ CO (/ (- CA CO) W))) O A))
      (set (read (strcat "O" P)) (/ (distance O A) W)))
    (setq OZ (if 2D OX OZ)
          W (if (equal (setq UV (U*V (mapcar '- X O) (mapcar '- Y O)))
                       '(0 0 0) Q0)
                " e Y están alineados."
                (if (and (not 2D)
                         (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                       (mapcar '* UV (mapcar '- Z O))) 0 Q0))
                    ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq Z (trans (if 2D (mapcar '+ O UV) Z) 1 0)
        Z-XY Y Y (trans Y 1 0))
  (setvar "CMDECHO" 0)
  (command "OSMODE" 0
           "ATTDIA" 0
           "EXPERT" 2
           "SCP" "3" O X Z-XY)
  (setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1)
        Z-XY (list (car Z) (cadr Z) 0) I (if (< (last Z) 0) -1 1)
        NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                       (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
        NORM-Z (equal Z-XY O Q0)
        BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (if (and NORM-XY NORM-Z)
      (progn
        (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                      (if ATREQ (append WWAA-1 (if OO-4 () W)))))
        (if OO-4 (INSERT** BL OX OY (* OZ I) 0)))
      (INSERT*))
  (command "SCP" "PR"
           "INSNAME" BL
           "EXPERT" EXPERT
           "ATTDIA" ATDIA
           "OSMODE" OSN)
  (setvar "CMDECHO" ECO)
  (princ))

(defun ENTPRECEDING ()
  (setq A (entnext) K A)
  (while (not (equal K E))
         (setq A K K (entnext K)))
  A)
```

```
(defun C:DESCOMPOK (/ Q0 ECO BLOC* E LE E* SA A N K NEXTE)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D (si hay atributos ")
  (prompt "con caracteres")
  (prompt "\noblicuos o no situados en su plano base, la orden DESCOMP no lo ")
  (prompt "resolverá bien)")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              BLOC* (if (wcmatch BLOC* "ATRIBATIP_#*")
                        (progn
                          (setq NEXTE E)
                          (while (or (/= (cdr (assoc 0 (setq E (ENTPRECEDING)
                                                               LE (entget E))))
                                         "INSERT")
                                     (if (wcmatch (cdr (assoc 2 LE))
                                                  "ATRIBATIP_#*")
                                         (setq NEXTE E))))
                          (cdr (assoc 2 LE)))
                        BLOC*)
              ECO (getvar "CMDECHO"))
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (if (and (wcmatch BLOC* (strcat "*_" N ",*_" N N ",*_AMB_ATRIBSTIPS"))
                 (setq SA (ssget "P")))
            (progn
              (setq NEXTE (if NEXTE NEXTE (entnext E))
                    E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))) K -1)
              (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                     (REST-OBLIC))))
        (setq K 0)
        (while (and (setq E NEXTE)
                    (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                    (wcmatch (setq BLOC* (cdr (assoc 2 LE)))
                             (setq A (strcat "ATRIBATIP_" (itoa K) "_DE_*"))))
          (command "DESCOMP" E ())
          (setq NEXTE (entnext E) E (ssname (ssget "P") 0)
                E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*)))
                E* (if (wcmatch BLOC* (strcat A "_" N)) (entnext E*) E*))
          (REST-OBLIC))
        (setvar "CMDECHO" ECO))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

Poc abans del codi precedent, advertíem al lector d'un desajust residual i menor, provocat per la necessitat de limitar-nos a la informació continguda en els blocs substituts i auxiliars, abstenint-nos de beure de la font original **BLOC** mentre no se seguís el protocol d'acceptar-ne l'eventual redefinició (recordem que aquesta exigència derivava, al seu torn, de la necessitat d'usar també ***INS2D***/***INS3D*** per a insercions ortogonals de **BLOC** amb atributs atípics, cosa que convida a reflexionar sobre l'anomenat "efecte papallona" ...): la decisió de suprimir la característica *Verificable* ens duïa a la paradoxa d'una primera inserció de **BLOC** amb ***INS2D***/***INS3D*** (primera absoluta o primera post-redefinició acceptada) en què aquests atributs repetirien la demanda de valor, cosa que en insercions posteriors ja no succeïa. Subsanarem aquesta última deficiència i, havent assolit una relativa satisfacció en els temes principals, recapitularem i ens ocuparem també dels secundaris:

- Començarem reestructurant el dispositiu que sol·licita valors per als atributs, de manera que **VAL-ATRIBS-1** s'apliqui cada cop a un sol atribut i **SEGR-ATRIBS** la pugui integrar en l'anàlisi de components de **BLOC**, acabat el qual **VAL-ATRIBS-2** passarà a la segona tanda de respostes per als atributs verificables, com abans. Sols que ara la segona volta ja no estarà limitada a aquesta anàlisi, sinó que també s'estendrà als altres casos (**6**, **7**, **24**, **25**, **30** i **31**, seguint amb el mateix nomenclàtor), on **VAL-ATRIBS-2** anirà rematant la feina feta per **VAL-ATRIBS-1** dins de la funció **VAL-ATRIBS**.

- Als arguments de l'antiga **VAL-ATRIBS-1**, **OO** (llista d'atributs P i N processats) i **E1E2** (existència dels dos atributs Constants i Invisibles "WWAA-1" i "WWAA-2", que encapçalaven **BLOC-1** i on hi guardàvem els directoris **E1** i **E2** dels atributs típics i atípics si n'hi havia d'aquests últims), ara **VAL-ATRIBS** n'afegirà un tercer, **EXT**, per distingir els casos en què la informació s'ha pres directament de **BLOC** o n'és **EXT**erior. Quan sigui **EXT** i **E1E2**, i processem la llista de tots els atributs P i N (fussió de **OO-2** i **OO-3**, amb recuperació de l'ordre original), cosa que fem des de **BLOC->WWAA**, caldrà saber quants atributs típics Verificables hi ha, per afegir a la cua de **WWAA-1** un nombre igual de textos buits: per tal de no complicar més el procés, hem inclòs aquesta dada a la capçalera de **E1**.

- La preservació de la característica *Verificable* comportarà la reaparició d'un petit contratemps que arrossegàvem des de la VERSIÓ 2 i que crèiem haver deixat enrera amb la VERSIÓ 17+: l'extemporània repetició de l'avís *Verificar valores de atributos* sempre que **BLOC** tingui atributs d'aquesta mena, inconveniència que ara se'ns agrava perquè si són atípics es resoldran amb insercions addicionals (el missatge no desitjat apareixerà una vegada per al conjunt d'atributs típics verificables, i una més per cada atributs atípic amb aquesta característica).

- I ja que parlem de qüestions formals, no ens oblidem de reincorporar al codi les expressions que giren a l'entorn del dispositiu ***DESHACER Inicio*/*Fin***, incloses a la VERSIÓ 16+ però que havíem decidit ometre provisionalment, ja en el present capítol, per no fer encara més feixuc el discurs que han il·lustrat les versions 17+ i 18+. Però no us penséssiu pas que és un trasllat mecànic de continguts:
  - Potser la funció **PRO-DESHACER-I/F** en tindrà prou amb una simple transposició a la nova realitat: si abans l'existència d'un genèric **BLOC-2*** de contingut nul s'expressava amb la condició **OO-2**, ara la diversificació d'atributs editables (típics i atípics) requerirà la seva substitució per **(or OO-2 OO-4)**.
  - Però quan anem a reubicar **VAL-ATRIBS** (ara **VAL-ATRIBS-1** i **VAL-ATRIBS-2**), veurem que cal recórrer a l'alta cirurgia si volem preservar l'eficàcia de **getstring** com a via d'assignació per teclat dels valors d'atribut. Recordeu que aquesta porta d'entrada s'ha de situar abans de **(command ... DESHACER I ...)** per tal d'evitar que una introducció accidentada (per exemple, rectificant a temps els primers caràcters escrits) pugui convertir en paper mullat l'acció del tàndem ***DESHACER I*** / ***DESHACER F*** en relació a la neutralització integral de l'última ***INS2D*/*INS3D*** amb una simple ordre ***H***. Recapitulem:
    - En relació a la VERSIÓ 16+, a la VERSIÓ 17+ havíem permutat les posicions relatives de **(SEGR-ATRIBS)** i **(VATR)**: des d'aparèixer just en aquest ordre, **(VAL-ATRIBS)>(VATR)** passava a situar-se davant de **(SEGR-ATRIBS)**, i davant també de la seqüència

```
(setq Z (trans (if 2D (mapcar '+ O UV) Z) 1 0)
      Z-XY Y Y (trans Y 1 0))
(command "DESHACER" (progn ... "I"))
.....................................
(setvar "CMDECHO" 0)
(command ... "SCP" "3" O X Z-XY)
```

    - En relació a la VERSIÓ 17+, a la VERSIÓ 18+ havíem introduït **(VATR)** a dintre de **(SEGR-ATRIBS)**, on apareixía tres cops:
      - **(BLOC->WWWWA)>(VAL-ATRIBS-1 OORD T)>(VATR)**
      - **(VAL-ATRIBS-1 OO-2 ())>(VATR)**
      - **(VAL-ATRIBS-2)>(VATR)**

      però tots tres darrera de la seqüència esmentada, raó per la qual incomplia la condició d'eficàcia requerida.
    - En relació a la VERSIÓ 18+, en la present VERSIÓ 19+ **(VATR)** ha de mantenir-se a l'interior de **(SEGR-ATRIBS)**, on hauria d'aparèixer quatre cops:
      - **(BLOC->WWAA)>(VAL-ATRIBS OORD T T)>(VAL-ATRIBS-1 O)>(VATR)** i **>(VAL-ATRIBS-2)>(VATR)**
      - **(VAL-ATRIBS OO-2 T ())>(VAL-ATRIBS-1 O)>(VATR)** i **>(VAL-ATRIBS-2)>(VATR)**
      - **(VAL-ATRIBS-1 LE)>(VATR)**
      - **(VAL-ATRIBS-2)>(VATR)**

      I cal aconseguir que tots aquestes quatre crides es produeixin abans de la seqüència esmentada. Tal i com ho tenim, caldrà vèncer 2 obstacles:

      1- L'accés a **SEGR-ATRIBS** s'ha de fer sabent ja si s'acompleix **NORM-XY** i/o **NORM-Z**, però la determinació d'aquestes condicions la fèiem després de la seqüència referida, bàsicament perquè, sent inevitable realitzar les assignacions

```
(setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1)
      Z-XY (list (car Z) (cadr Z) 0) ...)
```
resultava més simple fer-ho tot seguit
```
(setq ... NORM-Z (equal Z-XY O Q0) ...)
```
que deixar aquestes assignacions per a més endavant i calcular
```
(setq NORM-Z (and (equal (expt (distance X Z) 2)
                         (+ (expt OX 2) (expt OZ 2)) Q0)
                  (equal (expt (distance Y Z) 2)
                         (+ (expt OY 2) (expt OZ 2)) Q0))
```
Però si no hi ha més remei que determinar el valor de **NORM-XY** i **NORM-Z** abans de les crides a **VATR** fetes des de **SEGR-ATRIBS**, i sols després d'aquestes crides podrem executar la seqüència
```
(setq Z (trans (if 2D (mapcar '+ O UV) Z) 1 0)
      Z-XY Y Y (trans Y 1 0))
(command "DESHACER" (progn ... "I"))
.....................................
(setvar "CMDECHO" 0)
(command ... "SCP" "3" O X Z-XY)
```
i les posteriors assignacions
```
(setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1) ...)
```
caldrà recórrer al segon procediment i convertir aquesta seqüència en una subrutina (l'anomenarem **INICOMMAND**) per no ser reiteratius en el codi, ja que no s'hi val a situar-la després de **SEGR-ATRIBS** i haurà de figurar en dos llocs diferents dintre d'aquesta funció: a **SEGR-ATRIBS** hi ha dos itineraris alternatius (o ja disposem dels blocs substituts precisos i només hem d'assignar valors als atributs, o encara ens cal fabricar-los), en cadascun dels quals intervenim en el dibuix amb les funcions **entmake** i **command**, i l'execució de ***DESHACER I*** ha de ser prèvia. A **INICOMMAND** s'hi accedirà des de dues posicions de **SEGR-ATRIBS**: si s'executa des de la primera posició, ho serà un cop s'hagi avaluat **(BLOC->WWAA)**>**(VAL-ATRIBS OORD T T)**>**(VAL-ATRIBS-1 O)**>**(VATR)** i >**(VAL-ATRIBS-2)**>**(VATR)**, o **(VAL-ATRIBS OO-2 T ())**>**(VAL-ATRIBS-1 O)**>**(VATR)** i >**(VAL-ATRIBS-2)**>**(VATR)**; altrament, si es fa des de la segona, ho serà un cop s'hagi avaluat repetidament **(VAL-ATRIBS-1 LE)**>**(VATR)** i s'hagi rematat avaluant **(VAL-ATRIBS-2)**>**(VATR)**. Heus-la aquí:
```
(defun INICOMMAND ()
  (setq Z (trans (if 2D (mapcar '+ O UV) Z) 1 0)
        Z-XY Y Y (trans Y 1 0))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
  (setvar "CMDECHO" 0)
  (command "OSMODE" 0
           "ATTDIA" 0
           "EXPERT" 2
           "SCP" "3" O X Z-XY)
  (setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1)
        Z-XY (list (car Z) (cadr Z) 0) I (if (< (last Z) 0) -1 1)))
```
2- Ja a la VERSIÓ 17+, entre les primeres línies de **SEGR-ATRIBS** havíem introduït el dispositiu
```
(if (< K OO-4)
    (setq OO-4 K)
    (if (> K OO-4)
        (progn
          (setq NL2EX () BL2EX ())
          (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))))
```
on **K** representava el nombre de tríades de punts definitòries d'atributs atípics localitzades a **NLOC-2** o **BLOC-2**, i **OO-4** el de blocs portadors amb numeració correlativa a partir de ATRIBATIP_1_DE_<nom_de_**BLOC**>. Per tal de sanejar situacions en què **(/= K OO-4)**, a causa de la vulnerabilitat del conjunt d'aquests blocs 2D a tot tipus de manipulacions externes, decidíem utilitzar el mateix remei que el mític Procust, igualitarista radical *avant la lettre*, aplicava a tots els infeliços que queien a les seves urpes, asserrant-los les cames si sobrepassaven la llargada del seu llit, o estirant-los-les si no hi arribaven. Si **(< K OO-4)**, no dubtàvem a fer **(setq OO-4 K)** per tal que en **INSERT*** fos ignorat l'excedent de blocs portadors més enllà de l'anomenat ATRIBATIP_<**K**>_DE_<nom_de_**BLOC**>, però el problema es presentava quan **(> K OO-4)**, perquè no ens podíem treure de la mànega més blocs portadors (ni era raonable destruir les tríades de punts

sobrants, informació més fiable que uns blocs que qualsevol podia haver eliminat). Enfrontats al dilema, optàvem per la destrucció de **NLOC-2** i **BLOC-2**, com a recurs més efectiu per defermar una reconstrucció completa dels blocs que mentre treballem amb ***INS2D***/***INS3D*** gestionen la informació continguda a **BLOC**. Dons bé, l'última línia del fragment de codi que hem transcrit és també l'últim escull a superar, perquè no hauría d'aparèixer mentre **INICOMMAND** no hagues estat executat. Sortosament, com que allò que de debò desencadena la reconstrucció integral no és aquesta última línia sinó la precedent, la intervenció de ***LIMPIA*** la diferirem i de moment ens limitarem a activar un senyal (la nova variable **FW**) perquè més endavant (després de **INICOMMAND**, en l'itinerari [re]constructor de **SEGR-ATRIBS**) la neteja es dugui a terme.

Aclarides les raons que ens han empès a revisar alguns aspectes de la VERSIÓ 18+ heus aquí complet el nou codi:

```
; VERSIÓ 19+

;;; (vl-load-com)
;;; (vlr-remove-all)
;;; (vlr-COMMAND-reactor ()'((:vlr-CommandWillStart . -ECO-1)
;;;                          (:vlr-CommandEnded . -ECO-2)))

;;; (defun -ECO-1 (N-REACTIU L-ORDRE)
;;;   (if CTRL--ECO (BS (1+ (strlen (getcname (strcat "_" (car L-ORDRE))))))))

;;; (defun -ECO-2 (N-REACTIU L-ORDRE)
;;;   (if CTRL--ECO (BS (1+ (strlen CTRL--ECO)))))

;;; (defun SENSE-RASTRE ()
;;;   (command "DESHACER" (progn (if (= ECO 1)
;;;                                  (progn
;;;                                    (BS 100)
;;;                                    (setq CTRL--ECO "I")))
;;;                              "I"))
;;;   (if (= ECO 1)
;;;       (progn
;;;         (BLANC)
;;;         (setq CTRL--ECO ()))))

(defun QUASI-SENSE-RASTRE ()
  (command "DESHACER" (progn (if (= ECO 1) (BS 100)) "I"))
  (if (= ECO 1) (BLANC)))

(defun C:INS2D () (INSERTOK T))

(defun C:INS3D () (INSERTOK ()))

(defun BS (I) (repeat I (princ "\10 \10")))

(defun BLANC ()
  (princ "\r") (repeat 100 (princ " ")) (princ "\r") (princ))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))
```

```
(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun U*V (U V)
  (list (- (* (cadr U) (last V)) (* (last U) (cadr V)))
        (- (* (last U) (car  V)) (* (car  U) (last V)))
        (- (* (car  U) (cadr V)) (* (cadr U) (car  V)))))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "") N M) (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS-1 (LA / N M V)
  (if (= (logand ICVP 8) 0)
      (progn
        (if (and ATREQ (not INI))
            (setq INI (not (prompt "\nIndique valores de atributo"))))
        (VATR (= (logand ICVP 4) 4) (cdr (assoc 2 LA))
              (cdr (assoc 3 LA)) (cdr (assoc 1 LA)))
        (setq LLAA (cons (list W N M V) LLAA)))))

(defun VAL-ATRIBS-2 (/ W1 N M V)
  (setq INI () W ())
  (foreach LA (reverse LLAA)
    (if (not EXT)
        (setq W1 (car LA)
              LA (cdr LA)))
    (if (car LA)
        (progn
          (if (and ATREQ (not INI))
              (setq INI (not (prompt "\nVerificar valores de atributo"))))
          (VATR (if (or W1 (and EXT (not E1E2))) (cons "" W) W)
                (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq V (list V))
    (if EXT
        (setq WWAA (append WWAA V))
        (if W1
            (setq WWAA-1 (append WWAA-1 V))
            (setq WWAA-2 (append WWAA-2 V)))))
  (if EXT
      (if E1E2 (repeat WW (setq W (cons "" W))))
      (setq E1 (strcat (itoa (length W)) E1)))
  (setq WW W))

(defun VAL-ATRIBS (OO EXT E1E2)
  (foreach O (reverse OO)
    (setq ICVP (cdr (assoc 70 O)))
    (if (and E1E2 (or (not O) (= (logand ICVP 8) 8)))
        (setq LLAA (cons () LLAA))
        (VAL-ATRIBS-1 O)))
  (VAL-ATRIBS-2))

(defun MAKEBLOC (BLOC/2 ATRIBS OO)
  (entmake (list '(0 . "BLOCK") (cons 2 BLOC/2) '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))
```

```
(defun MNLOC ()
  (MAKEBLOC (setq BLOC NLOC-1) (or AT-C OO-2) (append OO-2 OO-1))
  (MAKEBLOC NLOC-2 () OO-3))

(defun MBLOC ()
  (MAKEBLOC BLOC-1 AT-C OO-1)
  (MAKEBLOC BLOC-2 OO-2 (append OO-3 OO-2)))

(defun LINIA-BASE ()
  (polar (cdr (assoc 11 LE))
         (- (cdr (assoc 50 LE)) PI/2)
         (* (cdr (assoc 40 LE))
            (if (= C-74 1)
                (/ 1.0 -3)
                (if (= C-74 2) 0.5 1)))))

(defun PUNT (P) (list '(0 . "POINT") (cons 10 (trans P VZ 0)) (cons 210 VZ)))

(defun REST-OBLIC ()
  (setq A (cos (cdr (assoc 51 (entget E*)))))
  (if (not (equal A 1 Q0))
      (progn
        (setq LE (entget E)
              LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A))
                        (assoc 40 LE) LE)
              LE (subst (cons 41 (* (cdr (assoc 41 LE)) A))
                        (assoc 41 LE) LE))
        (entmod LE))))

(defun INSPARCIAL (EX EY EZ ANG / COMPROBLIC)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast))
        COMPROBLIC (not (= EX EY EZ)))
  (if (or OO-2 OO-4)
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast))
        (setq SA (ssget "P") K -1)
        (while (setq E (ssname SA (setq K (1+ K))))
               (ssadd E SS))))
  (if (and OO-4 (or NORM-Z COMPROBLIC))
      (progn
        (setq SA (ssadd) K (if NORM-Z (- (sslength SS) (* 3 OO-4)) 1))
        (repeat (* 3 OO-4)
                (setq E (ssname SS K))
                (ssadd E SA)
                (ssdel E SS))))
  (if COMPROBLIC
      (progn
        (setq K 0 E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC-2))))
        (while (setq K (1+ K) E (ssname SS K))
               (REST-OBLIC)
               (setq E* (entnext E*))))))

(defun XOR (A B) (and (not (and A B)) (or A B)))

(defun B->N (A B / C)
  (setq C (= B (if A BLOC-1 NLOC-1))
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E) AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if (and AT (or (and A (not C))
                             (and (not A) C
                                  (= (logand (cdr (assoc 70 LE)) 2) 0))))
                 (setq OO-2 (cons LE OO-2))
```

```
                (if C
                    (setq AT-C (if AT-C T AT)
                          OO-1 (cons LE OO-1))
                    (setq OO-3 (cons LE OO-3))))
            (setq E (entnext E)))))

(defun ATRIB-ATIPIC (I)
  (tblsearch "BLOCK"
             (strcat "ATRIBATIP_" (itoa (set I (1+ (eval I)))) "_DE_" BLOC)))

(defun BLOC->WWAA (/ EE OO EEOO EO OORD N)
  (setq OO (reverse OO-1)
        E1 (car OO) E2 (cadr OO)
        E1 (read (strcat "(" (cdr (assoc 1 E1)) ")"))
        WW (car E1) E1 (cdr E1)
        E2 (read (strcat "(" (cdr (assoc 1 E2)) ")")) N -1
        EE (repeat K (setq EE (cons (nth (setq N (1+ N)) E2) EE)))
        E2 (reverse EE)
        EE (append E1 E2) OO () N 0
        OO (repeat K
             (setq OO (cons (entget (cdr (assoc -2 (ATRIB-ATIPIC 'N)))) OO)))
        OO (reverse (append OO OO-2))
        EEOO (mapcar 'cons EE OO) N -1)
  (repeat (+ (length OO-2) J)
          (setq N (1+ N) EO (cdr (assoc N EEOO))
                OORD (cons EO OORD)))
  (VAL-ATRIBS OORD T T)
  (foreach E E1
           (if (setq N (nth E WWAA))
               (setq WWAA-1 (append WWAA-1 (list N)))))
  (foreach E E2
           (if (setq N (nth E WWAA))
               (setq WWAA-2 (append WWAA-2 (list N))))))

(defun WWAA->BLOC (/ O1 O2)
  (setq O1 (last OO-1))
  (if (and (= (cdr (assoc 0 O1)) "ATTDEF")
           (= (cdr (assoc 2 O1)) "WWAA-1")
           (= (cdr (assoc 70 O1)) 3))
      (setq OO-1 (reverse (cdr (cdr (reverse OO-1))))))
  (if OO-4
      (setq O1 (list '(0 . "ATTDEF") '(8 . "0") '(70 . 3) '(2 . "WWAA-1")
                     '(3 . "") (cons 1 E1) '(72 . 0) '(74 . 0) '(10 0 0 0)
                     '(210 0 0 1) '(7 . "Standard") '(40 . 1) '(41 . 1))
            O2 (subst (cons 1 E2) (cons 1 E1)
                      (subst '(2 . "WWAA-2") '(2 . "WWAA-1")
                             O1))
            OO-1 (append OO-1 (list O2 O1)))))

(defun INICOMMAND ()
  (setq Z (trans Z 1 0)
        Z-XY Y
        Y (trans Y 1 0))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
  (setvar "CMDECHO" 0)
  (command "OSMODE" 0
           "ATTDIA" 0
           "EXPERT" 2
           "SCP" "3" O X Z-XY)
  (setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1)
        Z-XY (list (car Z) (cadr Z) 0) I (if (< (last Z) 0) -1 1)))

(defun PRO-DESHACER-I/F ()
  (command "BLOQUE" BLOC-1* O (setq E (ssname SS 0)) "")
  (if (or OO-2 OO-4)
      (command "BLOQUE" BLOC-2* O (ssdel E SS) "")
      (MAKEBLOC BLOC-2* () ())))
```

```
(defun REDEF-BLOC*S (/ LB NZ)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (if BL1EX
      (progn
        (setq J (1+ (strlen BLOC-1))
              NZ NORM-Z NORM-Z ())
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC-1* (cdr (assoc 2 LB)))
                            (strcat BLOC-1 M (substr M 2)))
                   (progn
                     (setq E (cdr (assoc -2 LB))
                           LE (entget E)
                           K (cdr (assoc 41 LE))
                           BLOC-2* (strcat BLOC-2 (substr BLOC-1* J)))
                     (command "SCP" "EZ" O (cdr (assoc 210 LE)))
                     (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (command "SCP" "PR")
                     (PRO-DESHACER-I/F))))
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat N "*"))
                   (progn
                     (setq E (cdr (assoc -2 LB))
                           LE (entget E)
                           BLOC-1 (cdr (assoc 2 LE))
                           E (substr BLOC-1 J) NORM-Z (= E "")
                           BLOC-2 (strcat BLOC-2* E))
                     (if (not NORM-Z) (command "SCP" "EZ" O (cdr (assoc 210 LE))))
                     (INSPARCIAL (cdr (assoc 41 LE))
                                 (cdr (assoc 42 LE))
                                 (cdr (assoc 43 LE))
                                 (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (if (not NORM-Z) (command "SCP" "PR"))
                     (command "BLOQUE" BLOC* O SS "")
                     (if OO-4 (command "BLOQUE" (strcat "ATRIBSATIPS_DE_" BLOC*)
                                                O SA "")))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* NORM-Z NZ)))
  (if OO-4
      (progn
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2 OO-3 OO-2 OO-2 T NZ OO-4 OO-4 () J 0)
        (repeat NZ
                (setq J (1+ J) BLOC-2 (strcat "ATRIBATIP_" (itoa J) "_DE_" BL))
                (tblnext "BLOCK" T)
                (while (setq LB (tblnext "BLOCK"))
                       (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                                    (strcat BLOC-2 M))
                           (progn
                             (setq BLOC-1 "BLOC_NUL"
                                   E (cdr (assoc -2 LB))
                                   LE (entget E)
                                   K (cdr (assoc 41 LE)))
                             (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                             (command "BLOQUE" BLOC* O SS "")))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* OO-2 OO-3 OO-4 NZ)))
  (setvar "CLAYER" CAPA))

(defun J-PUNTS (LB)
  (setq E (cdr (assoc -2 LB)) J 0)
  (while E (setq J (if (= (cdr (assoc 0 (entget E))) "POINT") (1+ J) J)
                 E (entnext E)))
  J)
```

```
(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 NL1EX NL2EX BL1EX BL2EX AT AT-C NE
                      E1 E2 EXT INI LLAA C-10 C-11 C-72 C-74 OO-3 OZ VZ FW)
  (setq NORM (and NORM-XY NORM-Z)
        NLOC-1 (strcat BLOC "_AMB_ATRIBSTIPS")
        NL1EX (tblsearch "BLOCK" NLOC-1)
        NLOC-2 (strcat "ATRIBSATIPS_DE_" BLOC)
        NL2EX (tblsearch "BLOCK" NLOC-2)
        BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2) K 0
        OO-4 (while (ATRIB-ATIPIC 'K) K)
        K (if NL2EX
              (J-PUNTS NL2EX)
              (if BL2EX (J-PUNTS BL2EX)))
        K (if (> K 0) (/ K 3)) J OO-4)
  (if (< K OO-4)
      (setq OO-4 K)
      (if (> K OO-4) (setq NL2EX () BL2EX () FW T)))
  (if (and (or NL2EX BL2EX) (not (XOR NL1EX NL2EX)) (not (XOR BL1EX BL2EX)))
      (progn
        (if BL1EX
            (progn (B->N T BLOC-1) (B->N T BLOC-2))
            (progn (B->N () NLOC-1) (B->N () NLOC-2)))
        (if OO-3
            (BLOC->WWAA)
            (if OO-2 (progn
                       (VAL-ATRIBS OO-2 T ())
                       (setq WWAA-1 WWAA))))
        (INICOMMAND)
        (if (and (not NL1EX) NORM)
            (MNLOC)
            (if (not (or BL1EX NORM))
                (MBLOC)
                (if NORM (setq BLOC NLOC-1)))))
      (progn
        (setq OO-4 () NE -1 E1 "" E2 ""
              E (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
        (while E
               (setq LE (entget E))
               (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                        (= (logand (setq ICVP (cdr (assoc 70 LE))) 2) 0))
                   (progn
                     (setq NE (1+ NE)
                           C-72 (cdr (assoc 72 LE)) C-74 (cdr (assoc 74 LE))
                           LE (if (= C-72 4)
                                  (subst (cons 72 (setq C-72 1))
                                         (cons 72 4)
                                         (subst (cons 74 (setq C-74 2))
                                                (assoc 74 LE)
                                                LE))
                                  LE)
                           LE (if (and (< C-72 3) (> C-74 0))
                                  (subst (cons 74 0)
                                         (assoc 74 LE)
                                         (subst (cons 11 (LINIA-BASE))
                                                (assoc 11 LE)
                                                LE))
;;   Podeu usar les 3 línies precedents com alternativa a les 5 línies subsegüents
                                  (if (> C-72 0)                          ;;
                                      (subst (cons 11 (LINIA-BASE))       ;;
                                             (assoc 11 LE)                ;;
                                             LE)                          ;;
                                      LE))                                ;;
                                  LE)
                           OZ (last (assoc 10 LE))
                           VZ (cdr (assoc 210 LE)))
                     (VAL-ATRIBS-1 LE)
```

```
                    (if (and (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))
                        (setq LLAA (if (= (logand ICVP 8) 0)
                                       (cons (cons T (car LLAA)) (cdr LLAA))
                                       LLAA)
                              E1 (strcat E1 " " (itoa NE))
                              OO-2 (cons LE OO-2))
                        (setq LLAA (if (= (logand ICVP 8) 0)
                                       (cons (cons () (car LLAA)) (cdr LLAA))
                                       LLAA)
                              E2 (strcat E2 " " (itoa NE))
                              OO-3 (cons (PUNT (list 0 0 OZ)) OO-3)
                              OO-3 (cons (PUNT (list 1 0 OZ)) OO-3)
                              OO-3 (cons (PUNT (list 0 1 OZ)) OO-3)
                              C-10 (assoc 10 LE) C-11 (assoc 11 LE)
                              OO-4 (cons (subst (list 10 (cadr C-10)
                                                      (caddr C-10) 0)
                                                C-10
                                                (subst (list 11 (cadr C-11)
                                                             (caddr C-11) 0)
                                                       C-11
                                                       (subst '(210 0 0 1)
                                                              (assoc 210 LE)
                                                              LE)))
                                         OO-4)))))
                    (setq OO-1 (cons LE OO-1)
                          AT-C (if AT-C T AT)))
                (setq E (entnext E)))
          (VAL-ATRIBS-2)
          (INICOMMAND)
          (if FW (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))
          (if (or OO-4 (not NORM))
              (progn
                (command "REGENT"
                         "CECOLOR" "PORCAPA"
                         "CELTYPE" "PORCAPA"
                         "CELWEIGHT" -1)
                (WWAA->BLOC)
                (if (or NL1EX NL2EX NORM) (MNLOC))
                (if (or BL1EX BL2EX (not NORM)) (MBLOC))
                (if OO-4
                    (progn
                      (if (not (tblsearch "BLOCK" "BLOC_NUL"))
                          (MAKEBLOC "BLOC_NUL" () ()))
                      (setq J 0)
                      (foreach O (reverse OO-4)
                               (MAKEBLOC (strcat "ATRIBATIP_" (itoa (setq J (1+ J)))
                                                 "_DE_" BL) T (list O)))
                      (setq OO-4 (length OO-4))))
                (if (or BL1EX OO-4) (REDEF-BLOC*S))
                (setvar "CECOLOR" COL)
                (setvar "CELTYPE" TLIN)
                (setvar "CELWEIGHT" GLIN))))))

(defun CALCULA (U V OV AGUT U* U** OU* OU** V**)
  (setq A (polar O (- (angle O V) PI/2) OV)
        W (if AGUT (angle A U) (angle U A))
        B (mapcar '/ (mapcar '+ A U) '(2 2 2))
        W (angle O (polar B W (distance O B))))
  (set U* (inters U (polar U W 1) O B ()))
  (set U** (inters O (polar O (+ W PI/2) 1) U (eval U*) ()))
  (set OU* (distance O (eval U*)))
  (set OU** (distance O (eval U**)))
  (set V** (inters V (polar V W 1) O (eval U**) ())))

(defun FIX+ (O / N)
  (setq N (itoa (if (> (- O (setq N (fix O))) 0.5) (1+ N) N)))
  (repeat (- (strlen M) 1 (strlen N)) (setq N (strcat "0" N)))
  N)
```

```
(defun INSERT** (BLOC X Y Z G / JJ KK SSAA)
 (command "INSERT" (strcat "ATRIBSATIPS_DE_" BLOC) O "XYZ" X Y Z G
          "DESCOMP" (entlast)
          "SCP" "")
 (setq SSAA (ssget "P") JJ 0 KK -1)
 (repeat OO-4
         (setq JJ (1+ JJ) BLOC (strcat "ATRIBATIP_" (itoa JJ) "_DE_" BL)
               LE (entget (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
               ICVP (cdr (assoc 70 LE))
               WWAA-1 (if (= (logand ICVP 8) 0) (list (car WWAA-2)))
               WWAA-2 (if WWAA-1 (cdr WWAA-2) WWAA-2)
               WW (if (and WWAA-1 (= (logand ICVP 4) 4)) '(""))
               KK (1+ KK) O (cdr (assoc 10 (entget (ssname SSAA KK))))
               KK (1+ KK) X (cdr (assoc 10 (entget (ssname SSAA KK))))
               KK (1+ KK) Y (cdr (assoc 10 (entget (ssname SSAA KK)))))
         (command "SCP" "3" O X Y)
         (setq OX (distance O X) OY (distance O Y) OZ OX
               O '(0 0 0) X (list OX 0 0)
               Y (trans Y 0 1) Z (U*V X Y)
               Z-XY O I (if (< (last Z) 0) -1 1)
               BLOC-1 "BLOC_NUL" BLOC-2 BLOC
               NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                              (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
               NORM-Z T OO-2 T OO-4 ())
         (if NORM-XY
             (eval (append '(command "INSERT" BLOC O OX OY 0)
                           (if ATREQ (append WWAA-1 WW))))
             (INSERT*))
         (command "SCP" "PR"))
 (command "SCP" "PR"
          "BORRA" SSAA ""))

(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** Z-XY* XY XY* XY** OXY* OXY** ABLOC*)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X
            Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'OX* 'OX** 'Y**)
        (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
              A (inters Z-XY (polar Z-XY W 1) O X** ())
              Z-XY* (polar A (angle A Z-XY)
                           (/ (* (distance A Z-XY) (distance X* X**))
                              (distance X X**))))))
  (if NORM-Z
      (setq BLOC* (strcat BLOC "_"))
      (progn
        (setq X (inters X* Y* O Z-XY* ()) X (if X X Z-XY*)
              Z-XY (- (angle O X) (angle Y* X*))
              Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
              Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
              Z (trans (list (car Z-XY*) (cadr Z-XY*) (last Z)) 1 0))
        (command "SCP" "Z" O X
                 "SCP" "X" (* I 90))
        (setq XY (list OX* 0 0) Z (trans Z 0 1))
        (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'OXY* 'OXY** 'Z**)
        (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                            (FIX+ (/ OXY** OXY* Q0)))
              BLOC* (strcat BLOC K)
              BLOC-1* (strcat BLOC-1 K)
              BLOC-2* (strcat BLOC-2 K)
              BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
```

```
        (command "SCP" "PR" "SCP" "PR")
        (if (not BL*EX)
            (progn
              (INSPARCIAL (setq K (/ OXY* OXY**)) K K X*)
              (command "SCP" "Z" O X
                       "SCP" "X" 90
                       "GIRA" SS "" O XY*
                       "SCP" "Z" O XY**)
              (PRO-DESHACER-I/F)
              (command "SCP" "PR")))
        (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*
              BLOC* (strcat BLOC* "_" (FIX+ (/ ZZ** OXY** Q0))))))
  (if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
  (if OO-4 (setq ABLOC* (strcat "ATRIBSATIPS_DE_" BLOC*)))
  (if (or NORM-Z BL*EX)
      (setq BL*EX (and (tblsearch "BLOCK" BLOC*)
                       (if OO-4 (tblsearch "BLOCK" ABLOC*) T))))
  (if (not BL*EX)
      (progn
        (if NORM-Z
            (INSPARCIAL (setq K (/ OX* OX**)) K K X*)
            (progn
              (if B (command "SCP" "Z" O X "SCP" "X" 90))
              (INSPARCIAL (setq J (/ 1 OX**) K (* OXY** J))
                          (* (distance Z Z**) J) (* K (/ OX* OXY*)) XY**)
              (command "SCP" "PR" "SCP" "PR")))
        (command "SCP" "Z" O X**
                 "BLOQUE" BLOC* O SS "")
        (if OO-4 (command "BLOQUE" ABLOC* O SA ""))
        (command "SCP" "PR")))
  (command "CELWEIGHT" GLIN
           "CELTYPE" TLIN
           "CECOLOR" COL
           "CLAYER" CAPA)
  (eval (append '(command "INSERT" BLOC* O "XYZ" OX** (setq J (distance Y Y**))
                          (setq K (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1)
                                     OX** I)) X**)
                (if ATREQ (append WWAA-1 WW))))
  (if OO-4 (INSERT** BLOC* OX** J K X**)))

(defun INSERTOK (2D / Q0 2*PI PI/2 M N CAPA COL TLIN GLIN ECO CTRL--ECO OSN ATDIA
                      ATREQ EXPERT BL BLOC BLOC* BLOC-1 BLOC-2 BLOC-1* BLOC-2* UV
                      BL*EX ICVP WWAA WWAA-1 WWAA-2 WW SA SS O X Y Z Z-XY OX OY OZ
                      X-Y X-O-Y A B E LE E* I J K W NORM-XY NORM-Z OO-1 OO-2 OO-4)
  (setq Q0 0.001 M "_")
  (repeat (strlen (itoa (fix (/ 0.1 Q0))))
          (setq M (strcat M "#")))
  (setq 2*PI (* PI 2)
        PI/2 (/ PI 2)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        EXPERT (getvar "EXPERT")
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit))
        N (strcat BLOC M) W T)
```

```
  (while W
    (setq O (getpoint "\nPrecise punto de inserción: "))
    (foreach P (if 2D '("X" "Y") '("X" "Y" "Z"))
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                  " desde el punto de inserción: "))
            W (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC  " <1>: "))
            W (if W W 1))
      (set (read P) (mapcar '(lambda (CO CA) (+ CO (/ (- CA CO) W))) O A))
      (set (read (strcat "O" P)) (/ (distance O A) W)))
    (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))
          Z (if 2D (mapcar '+ O UV) Z) OZ (if 2D OX OZ)
          W (if (equal UV '(0 0 0) Q0)
                " e Y están alineados."
                (if (and (not 2D)
                         (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                       (mapcar '* UV (mapcar '- Z O))) 0 Q0))
                    ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq I (expt OX 2) J (expt OY 2) K (expt OZ 2)
        NORM-XY (equal (setq X-Y (expt (distance X Y) 2)) (setq X-O-Y (+ I J)) Q0)
        NORM-Z (or 2D (and (equal (expt (distance X Z) 2) (+ I K) Q0)
                           (equal (expt (distance Y Z) 2) (+ J K) Q0))
        BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (if (and NORM-XY NORM-Z)
      (progn
        (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                      (if ATREQ (append WWAA-1 WW))))
        (if OO-4 (INSERT** BL OX OY (* OZ I) 0)))
      (INSERT*))
  (command "SCP" "PR"
           "INSNAME" BL
           "EXPERT" EXPERT
           "ATTDIA" ATDIA
           "OSMODE" OSN
           "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))

(defun ENTPRECEDING ()
  (setq A (entnext) K A)
  (while (not (equal K E)) (setq A K K (entnext K)))
  A)

(defun C:DESCOMPOK (/ Q0 ECO BLOC* E LE E* SA A N K NEXTE)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D (si hay atributos ")
  (prompt "con caracteres")
  (prompt "\noblicuos o no situados en su plano base, la orden DESCOMP no lo ")
  (prompt "resolverá bien)")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              BLOC* (if (wcmatch BLOC* "ATRIBATIP_#*")
                        (progn
                          (setq NEXTE E)
                          (while (or (/= (cdr (assoc 0 (setq E (ENTPRECEDING)
                                                             LE (entget E))))
                                         "INSERT")
                                     (if (wcmatch (cdr (assoc 2 LE))
                                                  "ATRIBATIP_#*")
                                         (setq NEXTE E))))
                          (cdr (assoc 2 LE)))
                        BLOC*)
```

```
            ECO (getvar "CMDECHO"))
        (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
      ;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (if (and (wcmatch BLOC* (strcat "*_" N ",*_" N N ",*_AMB_ATRIBSTIPS"))
                 (setq SA (ssget "P")))
            (progn
              (setq NEXTE (if NEXTE NEXTE (entnext E))
                    E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))) K -1)
              (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                     (REST-OBLIC))))
        (setq K 0)
        (while (and (setq E NEXTE)
                    (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                    (wcmatch (setq BLOC* (cdr (assoc 2 LE)))
                             (setq A (strcat "ATRIBATIP_" (itoa K) "_DE_*"))))
          (command "DESCOMP" E ())
          (setq NEXTE (entnext E)
                E (ssname (ssget "P") 0)
                E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*)))
                E* (if (wcmatch BLOC* (strcat A "_" N)) (entnext E*) E*))
          (REST-OBLIC))
        (command "DESHACER" "F")
        (setvar "CMDECHO" ECO))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

Però no ens adormim, que encara queda un últim escull per superar. Perquè, quan en el paràgraf inicial del capítol concloïem que únicament els atributs horitzontals (paral·lels al pla ***XY*** del ***SCP*** vigent en crear el bloc) quedaven ben situats en inserir-los amb ***INSERT***, amb ***INS2D*** o amb ***INS3D*** en el cas particular **NORM-Z**, i que només els atributs horitzontals pertanyents al mateix pla que el punt base de **BLOC** quedaven ben situats en inserir-los amb ***INS3D*** si no es donava la condició **NORM-Z**, perquè els pertanyents a d'altres plans horitzontals patien un desplaçament tot i mantenir-se en el pla horitzontal on s'havien de situar, i la resta (tots els no horitzontals) anaven a parar al pla horitzontal que passava pel punt d'inserció, en rigor haguéssim hagut d'afegir: sempre que la informació relativa als atributs no constants de **BLOC**, emmagatzemada a la taula de blocs de la base de dades del dibuix, sigui correcta. Ara bé: ¿què volem dir exactament amb aquest condicional? ¿que potser la informació continguda a la base de dades no és fiable en qualsevol circumstància? Doncs no, perquè els blocs amb atributs N o P ubicats amb el mode de justificació ***M**edio* (amb el gairebé equivalent ***M**edio-**C**entro* no hi ha incidents) poden presentar deficiències en la descripció d'aquests components des del mateix moment de la seva constitució com a blocs, abans d'inserir-los i fer-se paleses les limitacions comentades. En teoria, el sistema de coordenades d'un bloc té els eixos para·lels als del ***SCP*** vigent en ser creat, amb l'origen desplaçat al punt base d'insercions, i el de cada atribut té l'eix **Z** orientat normalment al seu pla però tots comparteixen aquest mateix origen: el codi 10 representa el punt inicial de l'atribut en estat de preassignació (és a dir, amb un valor de text provisional que coincideix amb el seu nom identificador), i les seves coordenades haurien de respondre a aquest sistema; a diferència del codi 210, que representa el vector **VZ** unitari i està referit al sistema de coordenades del bloc. Doncs bé: tot i que en teoria el ***SCU**niversal* no hauria de pintar res en la descripció, quan el sistema de referència d'un bloc no sigui el ***SCU*** i el bloc tingui atributs N o P justificats amb ***M*** però no situats en cap pla coordenat ***SCU*** (o situats en algun d'aquests plans però sense que el punt base hi pertanyi), el ***SCU*** deixarà sentir la seva presència en les coordenades d'aquests atributs, amb uns codis 10 i 11 que no representen les posicions reals dels punts inicials i dels d'ancoratge.

Només quan el ***SCP*** vigent en executar ***BLOQUE*** sigui paral·lel al ***SCU*** i el punt base coincideixi amb l'origen d'aquest últim (és a dir, quan el sistema de referència del bloc coincideixi amb el ***SCU***) podrem assegurar que els codis 10 i 11 de tots els atributs N i P, incloent-hi els de justificació ***M***, són correctes. Altrament, caldrà recalcular aquestes posicions (n'hi hauria prou de fer-ho amb els codis 10) en tots els no horitzontals (codi 210 diferent de **'(0 0 1)**) o de coordenada **z** no nul·la en relació al ***SCU***, perquè depenem d'aquesta informació tant per ubicar directament els atributs (els situats en el pla **XY** del sistema de referència del

bloc, com s'ha dit) com per fer-ho mitjançant tríades de punts (tots els demés). Filant més prim, podríem eximir d'aquesta obligació els atributs pertanyents als plans coordenats **YZ** i **ZX**, sempre que el punt base del bloc se situés a l'eix **Z**, però tot plegat encara ho complicaría més. El pitjor és que la distorsió esmentada no només afectarà els resultats de ***INS3D***, sinó també els de ***INS2D*** i els de l'ordre bàsica ***INSERT***. Davant d'això no ni ha solucions simplistes, tret d'avisar l'usuari que en comptes de definir els atributs P i N amb el mode de justificació ***M*** ho faci amb ***MC***: **SEGR-ATRIBS** no podrà assumir una modificació de **BLOC** en aquesta línia, com ho fa amb **BLOC-2**, perquè per fer-ho necessita dades fiables que **BLOC** no garanteix.

Però el problema greu és que la descripció del bloc no conserva cap informació del ***SCP*** vigent quan es va crear el bloc: el codi 10 remet invariablement a la posició **'(0 0 0)** (almenys quan el codi 70 és menor que **4**, és a dir, quan no parlem d'una referència externa ni d'un bloc que en depengui) i no hi ha codi 210 (sempre serà **'(0 0 1)**, per definició). Com ho podríem fer, per disposar d'aquesta informació? Una possibilitat seria executar ***SCP Guardar*** tot just després de ***BLOQUE***, guardant el sistema de referència actual (per exemple, amb el mateix nom del bloc creat). Una altra, incorporar al bloc un nou atribut (podria ser *Predefinido* i *Invisible*, per no alterar les condicions d'inserció i de visualització, i hauria d'encapçalar el conjunt de components del bloc per facilitar-ne la detecció) per emmagatzemar la informació indispensable: origen i orientació **Z** del ***SCP*** vigent en crear el bloc (ambdues dades referides al ***SCU***), o bé origen i orientació **Z** del ***SCU*** (referides al ***SCP*** vigent en crear el bloc). I això es podria fer de dues maneres: com a valor de l'atribut (en format de text, que després caldria interpretar) o bé donant-li una justificació ***aLinea***r o ***aJustar***, i associant els punts **'(0 0 0)** i **'(0 0 1)** (de la diferència entre els quals n'obtindrem el vector **VZ** unitari) als codis 10 i 11. Descartat l'emmagatzament de ***SCP****s* amb nom, per la necessitat d'actualització que comportarien les redefinicions del bloc i les afectacions amb ***LIMPIA*** o ***RENOMBRA***, aquesta última opció sembla la més simple. La nova ordre ***BLOQUEOK***, alternativa de ***BLOQUE*** per incorporar automàticament aquest atribut d'emmagatzament d'informació geomètrica al conjunt d'objectes seleccionats, podria tenir l'aspecte següent:

```
; fragment de VERSIÓ 20+
(defun C:BLOQUEOK (/ Q0 BLOC I I1 I2 SS CAPA ECO EXPERT AFL)
  (prompt "\nImprescindible para definir un bloque con atributos editables ")
  (prompt "justificados")
  (prompt "\nen su punto Medio, si el SCP no es paralelo al SCU o el punto ")
  (prompt "base no es *0,0,0")
  (while (not BLOC)
         (setq BLOC (getstring "\nIndique nombre de bloque o [?]: " T))
         (while (= (substr BLOC 1 1) " ")
                (setq BLOC (substr BLOC 2)))
         (if (> (setq I (strlen BLOC)) 0)
             (while (= (substr BLOC I 1) " ")
                    (setq I (1- I) BLOC (substr BLOC 1 I))))
         (setq BLOC (if (wcmatch BLOC (strcat ",*[" (chr 1) "-" (chr 31)
                                              "]*,*\"*,*`**,*`,*," "*/*,*["
                                              (chr 58) "-" (chr 62)
                                              "]*,*?`?*,*`??*," "*\\*,*``*,*|*,*["
                                              (chr 127) "-" (chr 160) "]*"))
                        (prompt "\nNombre bloque no válido.")
                        BLOC)))
  (setq Q0 0.001 ECO (getvar "CMDECHO"))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
  (setvar "CMDECHO" 0)
  (if (= BLOC "?")
      (command "BLOQUE" "?" (progn
                              (prompt "\nIndique bloque(s) a enumerar <*>: ")
                              (setvar "CMDECHO" 1) PAUSE))
      (if (or (not (tblsearch "BLOCK" BLOC))
              (= "Si" (progn
                        (initget "Si No")
                        (getkword (strcat "\nEl bloque \"" BLOC "\" ya existe. "
                                          "¿Desea volver a definirlo? [Sí/No] <N>: "))))))
          (progn
            (setq CAPA (getvar "CLAYER")
                  EXPERT (getvar "EXPERT")
```

```
                AFL (getvar "AFLAGS")
                I (progn (initget 1)
                         (getpoint "\nPrecise punto base de inserción: "))
                I1 (trans '(0 0 0) 0 1) I2 (trans '(0 0 1) 0 1) SS (ssget))
          (command "CLAYER" "0"
                   "EXPERT" 2
                   "AFLAGS" 9
                   "CLAYER" "0"
                   "ATRDEF" "" "BLOQUEOK")
          (if (equal (list (car I1) (cadr I1)) (list (car I2) (cadr I2)) Q0)
              (command "" "" "NIN" I1 "" "")
              (command (rtos (- (last I2) (last I1)) 1 8) "" "J" "NIN" I1 "NIN"
                       (list (car I2) (cadr I2) (last I1)) ""))
          (command "BLOQUE" BLOC I (entlast) SS ""
                   "AFLAGS" AFL
                   "EXPERT" EXPERT
                   "CLAYER" CAPA))))
  (command "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))
```

Com haureu vist, no ha estat possible de seguir al peu de la lletra les intencions expressades més amunt, perqué els punts de codi 10 i 11 han de tenir la mateixa coordenada **z**: el que hem fet ha estat emmagatzemar sota el codi 11 les coordenades **x** i **y** de **I2** (vector **VZ** unitari ***SCU***) amb una **z** idèntica a la de **I1** (origen ***SCU***) i guardar la transcripció a text de la **z** del segon punt (en realitat, l'increment **z** del primer al segon) en un lloc discret com és el codi 3 (missatge de sol·licitud de valor, que en un atribut predefinit no s'arriba a utilitzar); això ens obliga a tractar de manera diversa (justificació simple a partir del punt inicial) el cas en què el ***SCP*** s'hagi obtingut del ***SCU*** per desplaçament en direcció **Z**, per no ser admissible que els dos punts de justificació ***aJustar*** coincideixin. Quant a **I1** i **I2**, no ha de sorprendre que ens haguem limitat a transformar-los al ***SCP*** vigent en executar ***BLOQUEOK***,

```
   (setq I (progn (initget 1)
                  (getpoint "\nPrecise punto base de inserción: "))
         I1 (trans '(0 0 0) 0 1)
         I2 (trans '(0 0 1) 0 1) ...)
```

en comptes de passar-los al sistema propi del bloc (origen en el punt base), fent

```
   (setq I (progn (initget 1)
                  (getpoint "\nPrecise punto base de inserción: "))
         I1 (mapcar '- (trans '(0 0 0) 0 1) I)
         I2 (mapcar '- (trans '(0 0 1) 0 1) I) ...)
```

La raó és que ja s'encarrega AutoCAD de fer la conversió addicional, en passar l'atribut-senyal **"BLOQUEOK"** a ser component del bloc.

Fins aquí, ens hem limitat a afirmar que les posicions representades pels codis 10 i 11 (punts inicial i d'ancoratge) d'atributs N i P amb justificació ***M*** no seran fiables si el sistema de referència del bloc no coincideix amb el ***SCU***, però no hem arribat a avaluar la magnitud de l'error, aventurant alguna hipòtesi que permeti definir l'acció i calcular la deformació (el vector diferència entre les posicions correctes i les obtingudes) en funció de la discordància de sistemes reflectida en els codis 10 i 11 de l'atribut "BLOQUEOK", inclòs el truquet que afecta el codi 3 (coordenades dels punts **'(0 0 0)** i **'(0 0 1)** ***SCU***, en el sistema propi del bloc).

La funció **REST-VALORS** ens servirà per assajar progressivament diverses hipòtesis. En una primera aproximació, ens hem limitat als blocs creats sota un ***SCP*** paral·lel al ***SCU***, i n'ha resultat que a les coordenades **z** dels punts inicial i d'ancoratge dels atributs sensibles a l'afectació se'ls suma la de l'origen ***SCU*** respecte al punt base. La correcció serà immediata, s'ubicarà en **SEGR-ATRIBS** i consistirà a restar-los-la: serà l'última coordenada del codi 10 de l'atribut "BLOQUEOK"; el codi 11 encara no el necessitem (en tot cas, el valor **'(0 0 1)** l'utilitzaríem per identificar la hipòtesi que ara contemplem, quan ampliéssim l'abast de l'estudi):

```
; fragment de VERSIÓ 20+
(defun AVIS ()
  (alert (strcat "ATENCIÓN:\n\nComo \"" BLOC "\" se creó con la orden BLOQUE "
                 "y no con BLOQUEOK,\n\nlos atributos justificados por su punto "
                 "Medio pueden aparecer movidos.")))
```

```
(defun AJUSTA-E (E)
  (if (= E "")
      ""
      (progn
        (setq OZ (mapcar '1+ (read (strcat "(" E ")"))) E "")
        (foreach Z OZ (setq E (strcat E " " (itoa Z)))))))

(defun REST-VALORS ()
  (setq VZ VZ OZ (- OZ (last (trans (car BLOK) '(0 0 1) VZ)))))

(defun REST-CODI ()
  (subst (list 10 (cadr C-10) (caddr C-10) 0)
         C-10
         (subst (list 11 (cadr C-11) (caddr C-11) 0)
                C-11
                (subst '(210 0 0 1)
                       (assoc 210 LE)
                LE))))

(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 NL1EX NL2EX BL1EX BL2EX BLOK AT AT-C NE
                      E1 E2 EXT INI LLAA C-10 C-11 C-72 C-74 PUNTM OO-3 OZ VZ)
  (setq NORM (and NORM-XY NORM-Z)
        NLOC-1 (strcat BLOC "_AMB_ATRIBSTIPS")
        NL1EX (tblsearch "BLOCK" NLOC-1)
        NLOC-2 (strcat "ATRIBSATIPS_DE_" BLOC)
        NL2EX (tblsearch "BLOCK" NLOC-2)
        BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2) K 0
        OO-4 (while (ATRIB-ATIPIC 'K) K)
        K (if NL2EX
              (J-PUNTS NL2EX)
              (if BL2EX (J-PUNTS BL2EX)))
        K (if (> K 0) (/ K 3))
        J OO-4)
  (if (< K OO-4)
      (setq OO-4 K)
      (if (> K OO-4) (setq NL2EX () BL2EX () FW T)))
  (if (and (or NL2EX BL2EX)
           (not (XOR NL1EX NL2EX)) (not (XOR BL1EX BL2EX)))
      (progn
        (if BL1EX
            (progn
              (B->N T BLOC-1)
              (B->N T BLOC-2))
            (progn
              (B->N () NLOC-1)
              (B->N () NLOC-2)))
        (if (and (setq LE (last OO-2))
                 (= (cdr (assoc 2 LE)) "NO-BLOQUEOK"))
            (AVIS))
        (if OO-3
            (BLOC->WWAA)
            (if OO-2
                (progn
                  (VAL-ATRIBS OO-2 T ())
                  (setq WWAA-1 WWAA))))
        (INICOMMAND)
        (if (and (not NL1EX) NORM)
            (MNLOC)
            (if (not (or BL1EX NORM))
                (MBLOC)
                (if NORM (setq BLOC NLOC-1)))))
      (progn
        (setq  E (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
               BLOK E OO-4 () NE -1 E1 "" E2 "")
```

```
        (while E
               (setq LE (entget E))
               (if (equal E BLOK)
                   (setq BLOK (if (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                       (= (cdr (assoc 2 LE)) "BLOQUEOK")) 1 0)))
               (if (= BLOK 1)
                   (setq C-72 (cdr (assoc 72 LE))
                         OZ (cdr (assoc 10 LE))
                         VZ (if (= C-72 0) '(0 0 1) (cdr (assoc 11 LE)))
                         VZ (if (= C-72 0)
                                VZ
                                (list (- (car VZ) (car OZ))
                                      (- (cadr VZ) (cadr OZ))
                                      (distof (cdr (assoc 3 LE)) 1)))
                         BLOK (list OZ VZ))
                   (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                            (= (logand (setq ICVP (cdr (assoc 70 LE))) 2) 0))
                       (progn
                         (setq NE (1+ NE)
                               C-72 (cdr (assoc 72 LE))
                               C-74 (cdr (assoc 74 LE))
                               PUNTM (= C-72 4)
                               BLOK (if (and (= BLOK 0) PUNTM)
                                        (progn
                                          (setq E1 (AJUSTA-E E1)
                                                E2 (AJUSTA-E E2)
                                                E1 (strcat " 0" E1)
                                                NE (1+ NE)
                                                OO-2
                                                  (append OO-2
                                                    (list
                                                      (list
                                                        '(0 . "ATTDEF")
                                                        '(8 . "0") '(70 . 9)
                                                        '(2 . "NO-BLOQUEOK")
                                                        '(3 . "") '(1 . "")
                                                        '(72 . 0) '(74 . 0)
                                                        '(10 0 0 0)
                                                        '(210 0 0 1)
                                                        (assoc 7 LE)
                                                        (assoc 40 LE)
                                                        (assoc 41 LE)))))
                                          (AVIS))
                                        BLOK)
                               LE (if (= C-72 4)
                                      (subst (cons 72 (setq C-72 1))
                                             (cons 72 4)
                                             (subst (cons 74 (setq C-74 2))
                                                    (assoc 74 LE)
                                                    LE))
                                      LE)
                               LE (if (and (< C-72 3) (> C-74 0))
                                      (subst (cons 74 0)
                                             (assoc 74 LE)
                                             (subst (cons 11 (LINIA-BASE))
                                                    (assoc 11 LE)
                                                    LE))
;;                                                                          ;;
;;                                                                          ;;
;;                                                                          ;;
;;   Podeu usar les 3 línies precedents com alternativa a les 5 línies subsegüents
                                             (if (> C-72 0)                 ;;
                                                 (subst (cons 11 (LINIA-BASE)) ;;
                                                        (assoc 11 LE)       ;;
                                                        LE)                 ;;
                                                 LE))                       ;;
                                      LE)
                               C-10 (assoc 10 LE)
                               C-11 (assoc 11 LE)
                               VZ (cdr (assoc 210 LE))
                               OZ (last C-10))
```

```
                        (if (and PUNTM BLOK) (REST-VALORS))
                        (VAL-ATRIBS-1 LE)
                        (if (and (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))
                            (setq LLAA (if (= (logand ICVP 8) 0)
                                           (cons (cons T (car LLAA)) (cdr LLAA))
                                           LLAA)
                                  E1 (strcat E1 " " (itoa NE))
                                  OO-2 (cons (if (and PUNTM BLOK) (REST-CODI) LE)
                                             OO-2))
                            (setq LLAA (if (= (logand ICVP 8) 0)
                                           (cons (cons () (car LLAA)) (cdr LLAA))
                                           LLAA)
                                  E2 (strcat E2 " " (itoa NE))
                                  OO-3 (cons (PUNT (list 0 0 OZ)) OO-3)
                                  OO-3 (cons (PUNT (list 1 0 OZ)) OO-3)
                                  OO-3 (cons (PUNT (list 0 1 OZ)) OO-3)
                                  OO-4 (cons (REST-CODI) OO-4))))
                    (setq OO-1 (cons LE OO-1)
                          AT-C (if AT-C T AT))))
            (setq E (entnext E)))
        (VAL-ATRIBS-2)
        (INICOMMAND)
        (if FW (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))
        (if (or OO-4 (not NORM))
            (progn
              (command "REGENT"
                       "CECOLOR" "PORCAPA"
                       "CELTYPE" "PORCAPA"
                       "CELWEIGHT" -1)
              (WWAA->BLOC)
              (if (or NL1EX NL2EX NORM) (MNLOC))
              (if (or BL1EX BL2EX (not NORM)) (MBLOC))
              (if OO-4
                  (progn
                    (if (not (tblsearch "BLOCK" "BLOC_NUL"))
                        (MAKEBLOC "BLOC_NUL" () ()))
                    (setq J 0)
                    (foreach O (reverse OO-4)
                      (MAKEBLOC (strcat "ATRIBATIP_" (itoa (setq J (1+ J)))
                                        "_DE_" BL) T (list O)))
                    (setq OO-4 (length OO-4))))
              (if (or BL1EX OO-4) (REDEF-BLOC*S))
              (setvar "CECOLOR" COL)
              (setvar "CELTYPE" TLIN)
              (setvar "CELWEIGHT" GLIN))))))
```

Observeu que ens hem tret de la màniga un altre atribut Invisible (ara Predefinit, no pas Constant) que no guardarà informació geomètrica, com "BLOQUEOK", només farà el paper de senyal semànticament oposat al primer (no és per atzar que l'anomenem "NO-BLOQUEOK") i marcarà el bloc auxiliar **BLOC-2** quan l'absència de "BLOQUEOK" en **BLOC** coincideixi amb la presència d'atributs sensibles: aquest senyal garantirà l'activació de l'alarma **AVIS** en les successives aplicacions de ***INS2D***/***INS3D*** a **BLOC**.

De la senyalització reguladora del trànsit se n'ocupa la variable **BLOK**:
- Si **BLOC** no s'ha de revisar en profunditat (per crear o recrear els substituts **BLOC-1**/**BLOC-2** i/o **NLOC-1**/**NLOC-2**):
  - Si el primer objecte de **BLOC-2** o **NLOC-2** és l'atribut-senyal "NO-BLOQUEOK", es dispara **AVIS**.
- Si **BLOC** s'ha de revisar en profunditat (podent només detectar la presència de "BLOQUEOK", no la d'un "NO-BLOQUEOK" que encara que hagués existit se situava en uns suplents **BLOC-2** o **NLOC-2** ja obsolets i que anem a reconstruir):
  - Si l'atribut "BLOQUEOK" existeix:
    - A la primera passada (primer component de **BLOC**) el detecta, assigna a **BLOK** el valor provisional **1** (just per saber que és la primera passada) i:
      - Obté **OZ** (origen del ***SCU*** en relació al ***SCP*** vigent en crear **BLOC**).
      - Obté **VZ** (orientació ***Z*** del ***SCU*** en relació al ***SCP*** vigent en crear **BLOC**).
      - Assigna a **BLOK** la llista **(list OZ VZ)**.
      - No el registra enlloc (ni a **OO-1**, ni a **OO-2**, ni a **OO-3** ni a **OO-4**).

    - A les altres passades (resta de components de **BLOC**) registra l'objecte a **OO-1** si no és cap atribut P o N, a **OO-2** si és un atribut típic, i a **OO-3** i **OO-4** si és un atribut atípic. En els dos últims casos, si l'atribut P o N està justificat amb la modalitat ***M***, **REST-VALORS** modifica **VZ** i **OZ**.
  - Si "BLOQUEOK" no existeix:
    - A la primera passada ho detecta i assigna a **BLOK** el valor **0**.
    - A totes les passades (primera i totes les demés):
      - Si l'objecte no és cap atribut P o N el registra a **OO-1**.
      - Si és un atribut P o N:
        - Si l'atribut està justificat amb la modalitat ***M*** i **BLOK** existeix:
          - Dispara **AVIS** i anul·la **BLOK**.
          - Incorpora l'atribut "NO-BLOQUEOK" a l'inici de **OO-2**)
        - El registra a **OO-2** si és típic, i a **OO-3** i **OO-4** si és atípic.

Pot semblar exagerat que llancem l'alarma *los atributos justificados por su punto Medio pueden aparecer movidos* només pel fet d'haver localitzat atributs normals o predefinits amb aquesta característica, sense l'atribut "BLOQUEOK" encapçalant la descripció dels components de **BLOC**: ¿que potser no havíem dit que mentre l'atribut i el punt base del bloc se situessin en el pla coordenat **XY** del ***SCU*** no hi hauria cap alteració dels codis 10 i 11? Sí, però això no autoritza a tenir més màniga ampla, perquè quan la tercera coordenada dels codis 10 i 11 sigui **0** i el codi 210 vagi seguit de **'(0 0 1)** ningú no ens garanteix que els dos primers no presenten valors alterats: n'hi ha prou a imaginar que un d'aquests atributs està situat en el pla **z** = **2** del ***SCU***, i que el bloc del qual forma part va ser creat sota aquest sistema, amb el punt base a una altura **z** = **1**.

Acabarem els comentaris sobre la definició de **SEGR-ATRIBS** que ocupa les 3 pàgines precedents amb una reflexió sobre la condició que hi figura subtratllada i que pot semblar sospitosa per excessivament simple. Res d'això: s'hi ha arribat després d'una acurada tria. Entre les solucions válides (en què només s'accepta que els atributs sensibles quedin mal situats quan **BLOC** no s'hagi fabricat amb ***BLOQUEOK***, és a dir quan sigui **(not BLOK)**) establirem categories segons el grau d'economía de recursos, prioritzant l'emmagatzament d'atributs en la llista **OO-2** (definicions formant part de **BLOC-2** o **NLOC-1**) per damunt de les llistes **OO-3** (tríades de punts formant part de **BLOC-2 o NLOC-2**) i **OO-4** (blocs 2D portadors individuals), sempre que parlem d'un atribut fatalment típic (típic de debò perquè no està justificat amb el punt ***M*** o perquè malgrat estar-ho **BLOK** li ha restituït la posició real, o que només ho és per error, perquè està justificat amb el punt ***M*** i és **(not BLOK)**):
- No és admissible usar, alternativament a la condició subratllada,
  (and (not PUNTM) (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))
  perquè quedaria a fora **(and PUNTM BLOK (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))**, i part dels atributs típics quedarien enregistrats a **OO-3** i **OO-4** com atípics.
- Si usem **(and (or (not PUNTM) BLOK) (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))** tots els atributs **(and PUNTM (not BLOK))** seran enregistrats a **OO-3** i **OO-4**, com atípics, fins i tot quan **(and (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))**, és a dir, quan hagin esdevingut erròniament típics.
- A l'altre extrem, acceptant que amb **(not BLOK)** els atributs **PUNTM** acabaran mal posats, **(or (and PUNTM (not BLOK))**
  (and (or (not PUNTM) BLOK) (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0)))
  ens podriem permetre enregistrar-los també a **OO-2**: fet i fet, si no hi ha manera d'ubicar-los correctament, que el cost sigui mínim.

Però aquest darrer argument és qüestionable: si no es pot evitar que els atributs se situïn fora de lloc, almenys que quedin ben orientats quan això sigui possible (el cas per al qual s'ha dissenyat la present **REST-VALORS**, en seria un exemple), en comptes de quedar-se tots paral·lels a la base del paral·lelepípede d'encaix. Entre els plantejaments oposats que representen les dues últimes opcions, la que hem incorporat al codi de **SEGR-ATRIBS** i hem subratllat garanteix que els atributs típics (els de ple dret i els que per les limitacions del sistema han anat a raure al mateix pla) quedaran registrats a la llista **OO-2**, i només els atípics *de facto* passaran a **OO-3** i **OO-4**.

A partir d'aquí podríem anar exigint cada cop més a **REST-VALORS**, passant a blocs amb el punt base situat a l'origen del ***SCU*** però creats sota un ***SCP*** no paral·lel, i aplegant finalment ambdues hipòtesis (absència de restriccions quant al punt base del bloc i el ***SCP*** vigent en crear-lo). Però, havent d'intervenir just en el moment de néixer **BLOC**, mitjançant una ordre alternativa a ***BLOQUE***, sembla més equànime no sobrecarregar ***INS2D***/***INS3D*** (tot i que solucionar el problema des de **SEGR-ATRIBS** no suposaria complicar-ne totes les execucions, sinó només la primera amb cada **BLOC**)

i desplaçar els càlculs a ***BLOQUEOK***. Des de **SEGR-ATRIB** ens limitarem a encapçalar **BLOC-2** amb l'atribut-senyal "NO-BLOQUEOK", quan l'absència en **BLOC** de "BLOQUEOK" (aquest reduït també a la condició d'atribut-senyal) coincideixi amb la presència d'atributs sensibles, per activar l'alarma **AVIS** en futures insercions obliqües, ara però sense cap funció **REST-VALORS** que intervingui en la definicó d'aquests atributs corregint la seva posició.

¿Com ens ho podem fer per crear un **BLOC** en les condicions escollides per l'usuari i sense guardar informació addicional que ***INS2D***/***INS3D*** hauria de processar després? Doncs canviant aquestes condicions però sense que l'usuari se n'adoni: definirem **BLOC** sobre una còpia dels objectes constitutius posicionada respecte al ***SCU*** tal i com ho estaven els originals en relació al que hauria d'haver estat el seu sistema de referència (el ***SCP*** actual, amb l'origen desplaçat al punt base); com que així, en coincidir ambdós sistemes, l'origen i el vector **VZ** unitari del ***SCU*** seran sempre **'(0 0 0)** i **'(0 0 1)**, ja no hi haurà cap informació geomètrica significativa amb què farcir "BLOQUEOK" i aquest atribut es limitarà a exercir de senyal per indicar a **SEGR-ATRIBS** si **BLOC** ha estat fabricat amb ***BLOQUEOK*** o amb l'ordre bàsica ***BLOQUE***. Alternativament, podem fer un ***BLOQUEOK*** que canvii de ***M*** a ***MC*** la justificació dels atributs N i P que haguem seleccionat com a components de **BLOC**: en aquest cas cal advertir l'usuari, perquè una cosa és la suplantació a **BLOC-2** (que només afectarà les insercions no trivials obtingudes amb ***INS2D***/***INS3D***, amb les deficiències que destacàvem en finalitzar el penúltim capítol) i una altra fer-la directament a **BLOC**, afectant qualsevol altre ús que en vulgui fer, fora de la nostra aplicació.

Pel que fa a la primera opció, moure la còpia dels components amb l'ordre ***ALINEAR*** (ordre no bàsica, sinó aportada per l'aplicació *geom3d.arx*), per fer coincidir el sistema de referència del futur bloc amb el ***SCU***, prometia ser tan immediat com bufar i fer ampolles, però hem hagut d'acabar fabricant la nostra pròpia eina: la funció **ALINEAR+**. Tot plegat consistia a considerar dos tetràedres trirectangles definits pels punts **'(0 0 0)**, **'(1 0 0)**, **'(0 1 0)** i **'(0 0 1)** dels dos sistemes de referència: el de **BLOC** (el ***SCP*** vigent en executar ***BLOQUEOK***, desplaçat fins al punt base) i el ***SCU***. Anomenant **PX**—**PY**—**PZ** i **QX**—**QY**—**QZ** els triangles equilàters dels dos sistemes (de $\sqrt{2}$ de costat), oposats als orígens respectius **OP** i **OQ**, només calia moure els components (i, amb ells, el primer tetràedre) fins a fer coincidir **PX** amb **QX**, **PY** amb **QY** i **PZ** amb **QZ** (després d'això, també **OP** coincidiria amb **OQ**), i ***ALINEAR*** semblava el mitjà més adient i senzill. Però les seves limitacions d'ús des d'AutoLISP, que no ens consta que estiguin publicitades, ens han fet desistir:

- La funció AutoLISP **alinear**, específica per accedir a l'ordre externa ***ALINEAR***, no ens anava bé perquè, en contra del que diuen els manuals, no admetia arguments per a la selecció d'objectes, com ara

```
(if (not (member "geom3d.arx" (arx))) (arxload "geom3d"))
(alinear "P" "" '(1 0 0) (trans '(1 0 0) 0 1)
                 '(0 1 0) (trans '(0 1 0) 0 1)
                 '(0 0 1) (trans '(0 0 1) 0 1))
```

  sinó que **"P"** s'havia d'ometre i fer la selecció en temps real.
- Si ignoràvem l'existència de **alinear** i tractàvem ***ALINEAR*** com una ordre bàsica, podíem avaluar fins a tres vegades seguides una expressió com

```
(if (not (member "geom3d.arx" (arx))) (arxload "geom3d"))
(command "ALINEAR" "P" "" "1,0,0" "*1,0,0"
                          "0,1,0" "*0,1,0"
                          "0,0,1" "*0,0,1")
```

  però la quarta execució de ***BLOQUEOK*** s'interrompia amb el missatge *ERROR FATAL: No se pueden anidar los comandos con más de 4 niveles*.

Així doncs, ha calgut improvisar una alternativa a ***ALINEAR***.

ALINEAR+ és una funció que, amb l'origen ***SCP*** situat ja en el punt base de **BLOC**, aplica als components (inclòs l'atribut-senyal "BLOQUEOK") els moviments següents:

- Els ***DESPLAZA*** des de **PX** fins a **QX**. En els tetràedres de referència, **PX** desplaçat coincideix amb **QX**.
- Els ***GIRA***, a l'entorn d'un eix normal al pla determinat pels punt **QX**, **QY** i el punt **PY** desplaçat (suposant que els costats **PX**—**PY** i **QX**—**QY** fossin paral·lels, el substituiríem pel punt **PZ** desplaçat) i que passa per **QX**, l'angle que forma el costat **PX**—**PY** (o **PX**—**PZ**) desplaçat amb el costat **QX**—**QY** . En els tetràedres de referència, **PX**—**PY** desplaçat i girat coincideix amb **QX**—**QY** .
- Els ***GIRA***, al voltant del costat **QX**—**QY** , l'angle que té com a vèrtex el punt mig d'aquest costat i que forma el punt **PZ**, desplaçat i girat, amb el punt **QZ**. Els tetràedres de referència coincidiran, en coincidir els triangles **PX**—**PY**—**PZ** i **QX**—**QY**—**QZ** , i **OP** amb **OQ** de retruc.

Naturalment, la traducció de tot plegat a algorisme es complica per la necessitat d'anar canviant de sistema de referència per tal que l'eix de gir tingui sempre la direcció **z** (l'ordre ***GIRA3D*** també és externa, com ***ALINEAR***!), i cada canvi arrossega al seu torn la necessitat de transformar les coordenades dels punts que haguem d'utilitzar, passant-les abans de l'antic ***SCP*** al ***SCU*** i després del ***SCU*** al nou ***SCP***.

Veiem ara el codi de **ALINEAR+** i **C:BLOQUEOK:**

```
; fragment de VERSIÓ 21A+
(defun ALINEAR+ (/ O PX PY PZ QX QY QZ D PYD PZD PYDU PZDU PZDXY PZDG PZDGU)
  (setq O '( 0 0 0)
        PX '(1 0 0)
        PY '(0 1 0)
        PZ '(0 0 1)
        QX (trans PX 0 1)
        QY (trans PY 0 1)
        QZ (trans PZ 0 1)
        D (mapcar '- QX PX)
        PYD (mapcar '+ PY D)
        PZD (mapcar '+ PZ D)
        PYDU (trans PYD 1 0)
        PZDU (trans PZD 1 0))
  (command "DESPLAZA" "P" "" PX QX
           "SCP" "3" QX QY (if (equal QY PYD Q0) PZD PYD))
  (setq PYD (trans PYDU 0 1) PZD (trans PZDU 0 1)
        PZDXY (list (car PZD) (cadr PZD) 0)
        PZDG (polar O (- (angle O PZDXY) (angle O PYD)) (distance O PZDXY))
        PZDG (list (car PZDG) (cadr PZDG) (last PZD))
        PZDGU (trans PZDG 1 0))
  (command "GIRA" "P" "" O "R" O PYD 0
           "SCP" "DE" (list (/ (sqrt 2) 2) 0 0)
           "SCP" "Y" 90
           "GIRA" "P" "" O "R" O (trans PZDGU 0 1) (trans PZ 0 1)
           "SCP" "PR" "SCP" "PR" "SCP" "PR"))

(defun C:BLOQUEOK (/ Q0 BLOC I SA SS ULT REDEF CAPA ECO OSN EXPERT AFL SCP)
  (prompt "\nImprescindible para definir un bloque con atributos editables ")
  (prompt "justificados")
  (prompt "\nen su punto Medio, si SCP no es paralelo al SCU o el punto base ")
  (prompt "no es *0,0,0.")
  (while (not BLOC)
         (setq BLOC (getstring "\nIndique nombre de bloque o [?]: " T))
         (while (= (substr BLOC 1 1) " ")
                (setq BLOC (substr BLOC 2)))
         (if (> (setq I (strlen BLOC)) 0)
             (while (= (substr BLOC I 1) " ")
                    (setq I (1- I) BLOC (substr BLOC 1 I))))
         (setq BLOC (if (wcmatch BLOC (strcat ",*[" (chr 1) "-" (chr 31)
                                              "]*,*\"*,*`**,*`,*," "*/*,*["
                                              (chr 58) "-" (chr 62)
                                              "]*,*?`?*,*`??*," "*\\*,*``*,*|*,*["
                                              (chr 127) "-" (chr 160) "]*"))
                        (prompt "\nNombre bloque no válido.")
                        BLOC)))
  (setq Q0 0.001
        ECO (getvar "CMDECHO"))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
  (setvar "CMDECHO" 0)
  (if (= BLOC "?")
      (command "BLOQUE" "?" (progn
                              (prompt "\nIndique bloque(s) a enumerar <*>: ")
                              (setvar "CMDECHO" 1) PAUSE))
      (if (or (not (tblsearch "BLOCK" BLOC))
              (= "Si" (progn
                        (initget "Si No")
                        (getkword (strcat "\nEl bloque \"" BLOC "\" ya existe. "
                                          "¿Desea volver a definirlo? [Sí/No] <N>: "))))))
```

```
        (progn
          (setq CAPA (getvar "CLAYER")
                OSN (getvar "OSMODE")
                EXPERT (getvar "EXPERT")
                AFL (getvar "AFLAGS")
                I (initget 1) I (getpoint "\nPrecise punto base de inserción: ")
                SA (ssget) SS (ssadd) ULT (entlast))
          (command "CLAYER" "0"
                   "OSMODE" 0
                   "EXPERT" 2
                   "AFLAGS" 9
                   "COPIA" SA "" "0,0,0" ""
                   "ATRDEF" "" "BLOQUEOK" "" "" I "" ""
                   "DESIGNA" (entlast) SA ""
                   "AFLAGS" AFL
                   "SCP" "DE" I)
          (while ULT (if (setq ULT (entnext ULT)) (ssadd ULT SS)))
          (ssdel (entlast) SS)
        ; Si conserves la línia precedent, en executar UY després de BLOQUEOK
        ; recuperaràs els objectes seleccionats. Si l'elimines o neutralitzes,
        ; a més d'això tindràs l'atribut "BLOQUEOK" en estat de preassignació.
          (setq SCP (= (getvar "WORLDUCS") 0))
          (if SCP (progn (ALINEAR+) (command "SCP" "")))
          (command "BLOQUE" BLOC "0,0,0" "P" "")
          (if SCP (command "SCP" "PR"))
          (command "SCP" "PR"
                   "BORRA" SS ""
                   "AFLAGS" AFL
                   "OSMODE" OSN
                   "CLAYER" CAPA))))
  (command "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))
```

A diferència de la versió precedent de **C:BLOQUEOK**, en què acabàvem amb un **(command "BLOQUE" ...)** (recordeu que, utilitzat com a argument de **command**, l'ordre ***BLOQUE*** es comporta com ***-BLOQUE*** i sempre ocasiona la pèrdua dels objectes designats com a components, que si convé podrem recuperar amb ***UY***), per provocar ara la il·lusió que executem ***-BLOQUE***, fent desaparèixer d'escena els objectes seleccionats, cal haver preparat les coses deliberadament perquè l'engany funcioni. Si haguéssim manipulat directament aquests objectes també s'autodestruirien amb ***-BLOQUE***, però en fer ***UY*** tindríem una sorpresa (almenys quan el sistema de referència del bloc no coincidís amb el ***SCU***): els hauríem recuperat però en una altra posició. Per evitar això, si haguéssim pogut servirnos de l'ordre externa ***ALINEAR*** la solució hagués estat senzilla: recuperar amb ***UY*** aquests objectes moguts, retornar-los al seu lloc d'origen usant per segona vegada ***ALINEAR*** (amb les posicions de partença i de destí permutades) i esborrar-les amb ***BORRA***. Però per estalviar-nos de crear una segona funció **ALINEAR+INVERSA**, hem preferit treure una còpia dels originals seleccionats: movem aquests originals (i no la còpia, com dèiem abans per simplificar), amb l'afegitó de l'atribut-senyal "BLOQUEOK", els convertim en bloc causant la seva autodestrucció i després esborrem la còpia (totes aquestes operacions, dintre de **C:BLOQUEOK**, per suposat); així, un ***UY*** posterior a ***BLOQUEOK*** recuperarà la còpia, que romania a la posició original. I, a propòsit de l'atribut-senyal "BLOQUEOK", hem d'aclarir que la característica *Invisible* que li hem conferit no es manifesta en estat de preassignació: això vol dir que, en fer ***UY*** després de ***BLOQUEOK***, no apareixerà el text "BLOQUEOK" (cosa que sí passava en la versió precedent, perquè usàvem els components originals i no una còpia desproveïda de l'atribut-senyal), però sí en inserir **BLOC** amb ***INSERT*** i explosionar-lo tot seguit amb ***DESCOMP***; a més, aquest text estarà situat en el punt d'inserció (no pas en l'origen ***SCU***, com en la versió precedent), circumstància que podem aprofitar com a test per saber *a priori* (abans d'aplicar-hi ***INS2D***/***INS3D*** i veure si sona o no l'alarma **AVIS**) si **BLOC** va ser creat amb ***BLOQUE*** o amb ***BLOQUEOK***.

Pel que fa a la versió alternativa que no precisa de **ALINEAR+**, tot seguit passem al codi perquè sobren explicacions: sols comentarem que respon a l'esquema de la VERSIÓ 20+, amb alguns elements de la precedent per tal que el resultat d'un ***UY*** immediat a ***BLOQUEOK*** ens restitueixi la matèria primera sense canvis (atributs de justificació ***M***, si s'escau) ni afegitons (senyal "BLOQUEOK"). Que l'usuari trii:

```
; fragment de VERSIÓ 21B+
(defun C:BLOQUEOK (/ Q BLOC I J K LK SA SS ULT CAPA ECO EXPERT AFL)
  (prompt "\nImprescindible para definir un bloque con atributos editables ")
  (prompt "justificados")
  (prompt "\nen su punto Medio, si SCP no es paralelo al SCU o el punto base ")
  (prompt "no es *0,0,0.")
  (while (not BLOC)
         (setq BLOC (getstring "\nIndique nombre de bloque o [?]: " T))
         (while (= (substr BLOC 1 1) " ") (setq BLOC (substr BLOC 2)))
         (if (> (setq I (strlen BLOC)) 0)
             (while (= (substr BLOC I 1) " ")
                    (setq I (1- I) BLOC (substr BLOC 1 I))))
         (setq BLOC (if (wcmatch BLOC (strcat ",*[" (chr 1) "-" (chr 31)
                                              "]*,*\"*,*`**,*`,*," "*/*,*["
                                              (chr 58) "-" (chr 62)
                                              "]*,*?`?*,*`??*," "*\\*,*``*,*|*,*["
                                              (chr 127) "-" (chr 160) "]*"))
                          (prompt "\nNombre bloque no válido.")
                          BLOC)))
  (setq Q T ECO (getvar "CMDECHO"))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
  (setvar "CMDECHO" 0)
  (if (= BLOC "?")
      (command "BLOQUE" "?" (progn
                              (prompt "\nIndique bloque(s) a enumerar <*>: ")
                              (setvar "CMDECHO" 1) PAUSE))
      (if (or (not (tblsearch "BLOCK" BLOC))
              (= "Si" (progn
                        (initget "Si No")
                        (getkword (strcat "\nEl bloque \"" BLOC "\" ya existe. "
                                          "¿Desea volver a definirlo? [Sí/No] <N>: "))))))
          (progn
            (setq CAPA (getvar "CLAYER")
                  EXPERT (getvar "EXPERT")
                  AFL (getvar "AFLAGS")
                  I (initget 1) I (getpoint "\nPrecise punto base de inserción: ")
                  SA (ssget) SS (ssadd) ULT (entlast) J -1)
            (command "COPIA" SA "" "0,0,0" "")
            (while ULT (if (setq ULT (entnext ULT)) (ssadd ULT SS)))
            (while (setq J (1+ J) K (ssname SA J))
                   (setq LK (entget K))
                   (if (and (= (cdr (assoc 0 LK)) "ATTDEF")
                            (= (cdr (assoc 72 LK)) 4)
                            (= (logand (cdr (assoc 70 LK)) 2) 0))
                       (progn
                         (if Q (setq Q (alert
                                         (strcat "ATENCIÓN:\n\nEn los atributos "
                                           "editables seleccionados,\n\nel modo de "
                                           "justificación M se sustituye por MC"))))
                         (entmod (subst (cons 72 1)
                                        (cons 72 4)
                                        (subst (cons 74 2)
                                               (assoc 74 LK)
                                               LK))))))
            (command "CLAYER" "0"
                     "EXPERT" 2
                     "AFLAGS" 9
                     "ATRDEF" "" "BLOQUEOK" "" "" I "" ""
                     "BLOQUE" BLOC I (entlast) SA ""
                     "BORRA" SS ""
                     "AFLAGS" AFL
                     "EXPERT" EXPERT
                     "CLAYER" CAPA))))
  (command "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))
```

Un cop detallades les dues versions alternatives de **C:BLOQUEOK** (en comptes de jugar amb el signe ";" a començament de línia, per neutralitzar-ne una i deixar l'altra, aconsellem d'incloure-les les dues posant en segon lloc la que es vulgui validar), prendrem VERSIÓ 19+ com a referència, i ens limitarem a reflectir els canvis, que només afecten a **SEGR-ATRIBS** (repetirem, doncs, **AVIS** i **AJUSTA-E**) i a **C:DESCOMPOK**. Pel que fa a **SEGR-ATRIBS**:

```
; fragment de VERSIÓ 21+
(defun AVIS ()
  (alert (strcat "ATENCIÓN:\n\nComo \"" BLOC "\" se creó con la orden BLOQUE "
                 "y no con BLOQUEOK,\n\nlos atributos justificados por su punto "
                 "Medio pueden aparecer movidos")))

(defun AJUSTA-E (E)
  (if (= E "")
      ""
      (progn
        (setq OZ (mapcar '1+ (read (strcat "(" E ")"))) E "")
        (foreach Z OZ (setq E (strcat E " " (itoa Z)))))))

(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 NL1EX NL2EX BL1EX BL2EX BLOK AT AT-C
                      NE E1 E2 EXT INI LLAA C-10 C-11 C-72 C-74 OO-3 OZ VZ FW)
  (setq NORM (and NORM-XY NORM-Z)
        NLOC-1 (strcat BLOC "_AMB_ATRIBSTIPS")
        NL1EX (tblsearch "BLOCK" NLOC-1)
        NLOC-2 (strcat "ATRIBSATIPS_DE_" BLOC)
        NL2EX (tblsearch "BLOCK" NLOC-2)
        BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2) K 0
        OO-4 (while (ATRIB-ATIPIC 'K) K)
        K (if NL2EX
              (J-PUNTS NL2EX)
              (if BL2EX (J-PUNTS BL2EX)))
        K (if (> K 0) (/ K 3)) J OO-4)
  (if (< K OO-4)
      (setq OO-4 K)
      (if (> K OO-4) (setq NL2EX () BL2EX () FW T)))
  (if (and (or NL2EX BL2EX) (not (XOR NL1EX NL2EX)) (not (XOR BL1EX BL2EX)))
      (progn
        (if BL1EX
            (progn
              (B->N T BLOC-1)
              (B->N T BLOC-2))
            (progn
              (B->N () NLOC-1)
              (B->N () NLOC-2)))
        (if (and (setq LE (last OO-2))
                 (= (cdr (assoc 2 LE)) "NO-BLOQUEOK"))
            (AVIS))
        (if OO-3
            (BLOC->WWAA)
            (if OO-2
                (progn
                  (VAL-ATRIBS OO-2 T ())
                  (setq WWAA-1 WWAA))))
        (INICOMMAND)
        (if (and (not NL1EX) NORM)
            (MNLOC)
            (if (not (or BL1EX NORM))
                (MBLOC)
                (if NORM
                    (setq BLOC NLOC-1)))))
      (progn
        (setq  E (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
               BLOK E OO-4 () NE -1 E1 "" E2 "")
```

```
        (while E
               (setq LE (entget E))
               (if (equal E BLOK)
                   (setq BLOK (if (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                       (= (cdr (assoc 2 LE)) "BLOQUEOK")) 2 0)))
               (if (> BLOK 1)
                   (setq BLOK 1)
                   (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                            (= (logand (setq ICVP (cdr (assoc 70 LE))) 2) 0))
                       (progn
                         (setq NE (1+ NE)
                               C-72 (cdr (assoc 72 LE)) C-74 (cdr (assoc 74 LE))
                               BLOK (if (and (= BLOK 0) (= C-72 4))
                                        (progn
                                          (setq E1 (AJUSTA-E E1)
                                                E2 (AJUSTA-E E2)
                                                E1 (strcat " 0" E1)
                                                NE (1+ NE)
                                                OO-2
                                                (append OO-2
                                                  (list
                                                    (list '(0 . "ATTDEF")
                                                          '(8 . "0")
                                                          '(70 . 9)
                                                          '(2 . "NO-BLOQUEOK")
                                                          '(3 . "")
                                                          '(1 . "")
                                                          '(72 . 0)
                                                          '(74 . 0)
                                                          '(10 0 0 0)
                                                          '(210 0 0 1)
                                                          (assoc 7 LE)
                                                          (assoc 40 LE)
                                                          (assoc 41 LE)))))
                                          (AVIS))
                                        BLOK)
                               LE (if (= C-72 4)
                                      (subst (cons 72 (setq C-72 1))
                                             (cons 72 4)
                                             (subst (cons 74 (setq C-74 2))
                                                    (assoc 74 LE)
                                                    LE))
                                      LE)
                               LE (if (and (< C-72 3) (> C-74 0))
                                      (subst (cons 74 0)
                                             (assoc 74 LE)
                                             (subst (cons 11 (LINIA-BASE))
                                                    (assoc 11 LE)
                                                    LE))
;;                                                                          ;;
;;                                                                          ;;
;;                                                                          ;;
;;   Podeu usar les 3 línies precedents com alternativa a les 5 línies subsegüents
                                             (if (> C-72 0)                 ;;
                                                 (subst (cons 11 (LINIA-BASE)) ;;
                                                        (assoc 11 LE)       ;;
                                                        LE)                 ;;
                                                 LE))                       ;;
                                      LE)
                               OZ (last (assoc 10 LE))
                               VZ (cdr (assoc 210 LE)))
                         (VAL-ATRIBS-1 LE)
                         (if (and (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))
                             (setq LLAA (if (= (logand ICVP 8) 0)
                                            (cons (cons T (car LLAA)) (cdr LLAA))
                                            LLAA)
                                   E1 (strcat E1 " " (itoa NE))
                                   OO-2 (cons LE OO-2))
                             (setq LLAA (if (= (logand ICVP 8) 0)
                                            (cons (cons () (car LLAA)) (cdr LLAA))
                                            LLAA)
```

```
                                E2 (strcat E2 " " (itoa NE))
                                OO-3 (cons (PUNT (list 0 0 OZ)) OO-3)
                                OO-3 (cons (PUNT (list 1 0 OZ)) OO-3)
                                OO-3 (cons (PUNT (list 0 1 OZ)) OO-3)
                                C-10 (assoc 10 LE) C-11 (assoc 11 LE)
                                OO-4 (cons (subst (list 10 (cadr C-10)
                                                        (caddr C-10) 0)
                                                  C-10
                                                  (subst (list 11 (cadr C-11)
                                                               (caddr C-11) 0)
                                                         C-11
                                                         (subst '(210 0 0 1)
                                                                (assoc 210 LE)
                                                                LE)))
                                           OO-4))))
                        (setq OO-1 (cons LE OO-1) AT-C (if AT-C T AT))))
                (setq E (entnext E)))
        (VAL-ATRIBS-2)
        (INICOMMAND)
        (if FW (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))
        (if (or OO-4 (not NORM))
            (progn
              (command "REGENT"
                       "CECOLOR" "PORCAPA"
                       "CELTYPE" "PORCAPA"
                       "CELWEIGHT" -1)
              (WWAA->BLOC)
              (if (or NL1EX NL2EX NORM) (MNLOC))
              (if (or BL1EX BL2EX (not NORM)) (MBLOC))
              (if OO-4
                  (progn
                    (if (not (tblsearch "BLOCK" "BLOC_NUL"))
                        (MAKEBLOC "BLOC_NUL" () ()))
                    (setq J 0)
                    (foreach O (reverse OO-4)
                      (MAKEBLOC (strcat "ATRIBATIP_" (itoa (setq J (1+ J)))
                                        "_DE_" BL) T (list O)))
                    (setq OO-4 (length OO-4))))
              (if (or BL1EX OO-4) (REDEF-BLOC*S))
              (setvar "CECOLOR" COL)
              (setvar "CELTYPE" TLIN)
              (setvar "CELWEIGHT" GLIN))))))
```

El nou plantejament ha retornat **SEGR-ATRIBS** a la simplicitat de VERSIÓ 19+, només alterada per la irrupció en escena dels atribut-senyal "BLOQUEOK" i "NO-BLOQUEOK". Ara, la regulació del trànsit a càrrec de **BLOK** respon al'esquema següent:

- Si **BLOC** no s'ha de revisar en profunditat (per crear o recrear els substituts **BLOC-1**/**BLOC-2** i/o **NLOC-1**/**NLOC-2**):
  - Si el primer objecte de **BLOC-2** o **NLOC-2** és l'atribut-senyal "NO-BLOQUEOK", es dispara **AVIS**.
- Si **BLOC** s'ha de revisar en profunditat:
  - Si l'atribut "BLOQUEOK" existeix:
    - A la primera passada (primer component de **BLOC**) el detecta, assigna a **BLOK** el valor provisional **2** (just per saber que és la primera passada) i:
      - Assigna a **BLOK** el valor **1**.
      - No el registra enlloc (ni a **OO-1**, ni a **OO-2**, ni a **OO-3** ni a **OO-4**).
    - A les altres passades registra l'objecte a **OO-1** si no és cap atribut P o N, a **OO-2** si és un atribut típic, i a **OO-3** i **OO-4** si és atípic.
  - Si "BLOQUEOK" no existeix:
    - A la primera passada ho detecta i assigna a **BLOK** el valor **0**.
    - A totes les passades (primera i totes les demés):
      - Si l'objecte no és cap atribut P o N el registra a **OO-1**.
      - Si és un atribut P o N:
        - Si l'atribut està justificat amb la modalitat ***M*** i **BLOK** existeix:
          - Dispara **AVIS** i anul·la **BLOK**.
          - Incorpora l'atribut "NO-BLOQUEOK" a l'inici de **OO-2**)
        - El registra a **OO-2** si és típic, i a **OO-3** i **OO-4** si és atípic.

Enllestida **SEGR-ATRIBS**, adaptar **C:DESCOMPOK** ens durà més feina que no semblava.

Si la descomposició amb ***DESCOMP*** d'un bloc creat mitjançant ***BLOQUEOK*** però inserit amb ***INSERT*** (perquè no tenía atributs atípics i sols volíem una inserció ortogonal) no arrossega altres imprevistos que la sorpresa del "BLOQUEOK" (sorpresa només si l'executor de ***DESCOMP*** n'ignorava l'origen), i fins i tot pot ser utilitzada com a test per saber si **BLOC** va ser creat amb ***BLOQUE*** o amb ***BLOQUEOK***, com hem suggerit en presentar la primera opció de ***BLOQUEOK***, no podem mirar amb la mateixa indulgència el cas simètric: aplicar ***DESCOMPOK*** a una inserció feta amb ***INS2D***/***INS3D***, obliqua o ortogonal (perquè hi havia atributs atípics, per exemple) però utilitzant un bloc definit amb ***BLOQUE*** i que té atributs justificats amb ***M***. Havent-lo fet amb ***BLOQUEOK*** no hi hauria cap problema, perquè l'atribut-senyal positiu "BLOQUEOK" s'annexa al **BLOC** original, mentre que ***INS2D***/***INS3D*** treballen amb els derivats **BLOC-1**/**BLOC-2** (o amb **NLOC-1**/**NLOC-2**) i els portadors d'atributs atípics. Però, fabricat amb ***BLOQUE*** i contenint atributs sensibles, la primera aplicació de ***INS2D***/***INS3D*** a **BLOC** detectarà aquestes circumstàncies i estigmatitzarà **BLOC-2** (o **NLOC-2**) amb l'atribut-senyal negatiu "NO-BLOQUEOK" que, com "BLOQUEOK", esdevé visible quan retorna a l'estat de preassignació, cosa que passa en descompondre la inserció. Podríem argumentar que, si malgrat el primer avís (que es repetirà cada cop que ***INS2D***/***INS3D*** utilitzi **BLOC**) i els possibles resultats deficients, l'usuari decideix de seguir endavant, ja no l'hauria d'agafar desprevingut aquesta fantasmagòrica aparició (que sempre serà a temps d'esborrar, si no li fa cap servei), però si pensem que ***BLOQUEOK*** no es indispensable (fins i tot en presència d'atributs sensibles), mentre ***BLOQUE*** actui sota el ***SCU****niversal* i els components se situïn de manera que el punt base coincideixi amb l'origen, no té sentit mantenir una pífia tan senzilla d'evitar. Disortadament, no trigarem a veure com els problemes no s'aturen aquí, però ja que aquest és el contratemps de detecció més immediata, ara ens limitarem a superar-lo incorporant a **C:DESCOMPOK** de la VERSIÓ 19+ el codi subratllat. Aquesta proposta provisional encaixaria en les versions 20+ i 21+ indistintament, però a mesura que avancem ens orientarem cap a la VERSIÓ 21+, i en concret cap a la variant 21B+:

```
(defun C:DESCOMPOK (/ Q0 ECO BLOC* E LE E* SA A N K NEXTE)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D (si hay atributos ")
  (prompt "con caracteres")
  (prompt "\noblicuos o no situados en su plano base, la orden DESCOMP no lo ")
  (prompt "resolverá bien).")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              BLOC* (if (wcmatch BLOC* "ATRIBATIP_#*")
                        (progn
                          (setq NEXTE E)
                          (while (or (/= (cdr (assoc 0 (setq E (ENTPRECEDING)
                                                               LE (entget E))))
                                         "INSERT")
                                     (if (wcmatch (cdr (assoc 2 LE))
                                                  "ATRIBATIP_#*")
                                         (setq NEXTE E))))
                          (cdr (assoc 2 LE)))
                        BLOC*)
              ECO (getvar "CMDECHO"))
        (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
      ;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (if (and (wcmatch BLOC* (strcat "*_" N ",*_" N N ",*_AMB_ATRIBSTIPS"))
                 (setq SA (ssget "P")))
            (progn
              (setq NEXTE (if NEXTE NEXTE (entnext E))
                    E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))) K -1)
              (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                     (if (and (= K 0)(= (cdr (assoc 2 (entget E))) "NO-BLOQUEOK"))
                         (entdel E)
                         (REST-OBLIC))))))
```

```
        (setq K 0)
        (while (and (setq E NEXTE)
                    (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                    (wcmatch (setq BLOC* (cdr (assoc 2 LE)))
                             (setq A (strcat "ATRIBATIP_" (itoa K) "_DE_*"))))
               (command "DESCOMP" E ())
               (setq NEXTE (entnext E)
                     E (ssname (ssget "P") 0)
                     E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*)))
                     E* (if (wcmatch BLOC* (strcat A "_" N)) (entnext E*) E*))
               (REST-OBLIC))
        (command "DESHACER" "F")
        (setvar "CMDECHO" ECO))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

Quan just abans de presentar aquesta versió de **C:DESCOMPOK** anunciàvem problemes, no ens referíem només a petits ajustos com el que podríem fer, substituint el tros de codi que inclou les línies que havíem subratllat

```
               (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                      (if (and (= K 0)
                               (= (cdr (assoc 2 (entget E))) "NO-BLOQUEOK"))
                          (entdel E)
                          (REST-OBLIC)))))
```

per

```
               (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                      (if (and (< K 2)
                               (= (cdr (assoc 2 (entget E))) "NO-BLOQUEOK"))
                          (entdel E)
                          (REST-OBLIC)))))
```

o per

```
               (setq EZ (= (cdr (assoc 0 (entget (ssname SA 0)))) "INSERT")
                     K (if EZ 0 -1))
               (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                      (if (and (= K (if EZ 1 0))
                               (= (cdr (assoc 2 (entget E))) "NO-BLOQUEOK"))
                          (entdel E)
                          (REST-OBLIC)))))
```

a fi d'abastar els casos en què **BLOC-2** incorpora l'atribut-senyal "NO-BLOQUEOK" i el conjunt d'objectes resultant d'explosionar la inserció de **BLOC**** no comença per aquest component sinó per la inserció de **BLOC-1***, que en les condicions **NORM-XY** no experimenta el retrocés a què hem fet esment més d'un cop, sinó d'altres supòsits en què ya no té sentit particularitzar i seguir complicant el codi. Per exemple, si apliquem ***DESCOMPOK*** a insercions ortogonals realitzades amb ***INS3D*** (**NORM-XY** i **NORM-Z**) encara tindrem fracassos estrepitosos, malgrat que el perfil dels blocs a explosionar sigui **(wcmatch BLOC* (strcat "*_" N ",*_" N N ",*_AMB_ATRIBSTIPS"))** i inclogui en últim lloc el nom dels substituts **NLOC-1**, cosa que expressa més un desig que una eficàcia garantida. Com sabeu, hi ha dues circumstàncies en què una inserció ortogonal trascendirà l'ús directe de **BLOC** i haurà de recórrer al tàndem **NLOC-1**/**NLOC-2:** quan **BLOC** tingui atributs atípics (encara que sigui el primer cop que ***INS3D*** l'utilitza) o quan **BLOC** ja hagi estat processat per ***INS3D*** en insercions obliqües (encara que no tingui atributs atípics). Doncs bé, en la primera sempre es produirà una interrupció i el resultat dependrà de l'origen de **BLOC:** si es va crear amb ***BLOQUEOK*** persistiran els dos atributs constants que guarden informació sobre la primitiva ordenació dels editables (abans de ser classificats en típics i atípics), "WWAA-1" i "WWAA-2"; si es va crear amb ***BLOQUE***, a més d'aquests també romandrà visible l'atribut-senyal "NO-BLOQUEOK". Pel que fa a la segona, només si hi ha atributs sensibles (ubicats amb el mode de justificació ***M****edio*) s'interromprà l'avaluació i quedarà sense esborrar l'atribut senyal "NO-BLOQUEOK". Però és que no n'hi ha prou a assegurar la correcta descomposició de **NLOC-1:** cal fer-ho també amb les insercions simples de **BLOC**, tant si resulten de l'aplicació de ***INSERT*** a blocs creats amb ***BLOQUE*** o amb ***BLOQUEOK*** (cas aquest en què cal esborrar l'atribut-senyal "BLOQUEOK"), com si ho són de l'aplicació de ***INS3D***, amb un paral·lelepípede ortogonal d'encaix, perquè explosionar-les amb ***DESCOMPOK*** és l'únic recurs per tal que atributs escrits amb caràcters oblics, aixafats en inserir el bloc portador, recobrin la seva alçada. Un cop més tapem forats, en el sentit que l'aplicació en què treballem no només amplia prestacions sinó que corregeix deficiències.

Què passa doncs quan explosionem la inserció simple d'un bloc compost per atributs editables i objectes d'altra mena (que no siguin blocs anònims, irreductibles)? Tret que el bloc hagués estat inserit amb escala uniforme ($\mathbf{E_x = E_y = E_z}$), dintre del conjunt resultant de l'expressió **(ssget "P")** passa que els atributs editables s'agrupen a continuació dels altres objectes, que al seu torn passen a situar-se en primer lloc. Dintre de cada grup, però, cada objecte manté la posició relativa que tenia en el bloc. Això no ens hauria de venir de nou, perquè no és més que una altra manifestació concreta del comportament genèric dels blocs explosionats, que ja coneixíem d'altres ocasions: ens l'havíem trobat a la VERSIÓ 14+ (a propòsit de la funció **PASSA-ATRIB**) i després, a la VERSIÓ 17+ havia tornat a aparèixer (aquest cop a propòsit de **INSPARCIAL**, explosionant insercions de **BLOC-2** o de **BLOC-2*** fetes amb $\mathbf{E_y \neq E_x}$ i/o $\mathbf{E_z \neq E_x}$, en què a més d'atributs típics hi havia objectes punt). Reproduïm les conclusions que n'havíem tret: l'explosió es propagava a tots els nivells en què fos possible (insercions amb escalat uniforme o polilínies); els components irreductibles que haguessin patit cisallament (insercions de blocs públics esdevingudes insercions de blocs anònims, o atributs en què l'obliqüitat hagués variat) passaven a l'últim lloc en el conjunt resultant de l'explosió, tot i que en la descripció del bloc explosionat no ocupessin aquesta posició. Just a continuació precisàvem que en aquesta reubicació per retrocés hi havia prioritats, segons el tipus d'objecte: per exemple, els blocs anònims passaven darrera dels atributs amb canvi d'obliqüitat. Això de "canvi d'obliqüitat" ja havíem precisat prèviament (en el capítol precedent, muntant la VERSIÓ 16+) que s'havia d'entendre en el sentit ampli de la variable **COMPROBLIC**: que amb qualsevol de les condicions $\mathbf{E_x \neq E_y}$ i $\mathbf{E_x \neq E_z}$ els atributs escrits amb caràcters oblics resultarien aixafats.

Totes les insercions esmentades en el penúltim paràgraf són susceptibles de ser explosionades amb ***DESCOMPOK***, malgrat la seva ortogonalitat, tant per restituir als atributs escrits amb caràcter oblics l'alcada original com per desempallegar-nos de l'atribut-senyal "BLOQUEOK" (en el cas d'haver inserit amb ***INSERT*** –o amb ***INS3D***, si tots els atributs eren típics i les insercions prèvies també eren ortogonals- un bloc creat amb ***BLOQUEOK***) o "NO-BLOQUEOK" (en el cas d'haver inserit amb ***INS3D*** un bloc amb atributs sensibles, creat ***BLOQUE***), o dels atributs auxiliars "WWAA-1" i "WWAA-2" (en el cas d'haver inserit amb ***INS3D*** un bloc amb atributs atípics), i tanmateix cap d'aquestes possibilitats no estava contemplada en l'última versió. I si volem incorporar-les no n'hi haurà prou amb intervencions puntuals (pedaços, en podríem dir), com fins ara, perquè les lectures paral·leles (sobre la definició del bloc explosionat i sobre el conjunt d'objectes resultants de l'explosió) dels atributs editables, que tenien per objecte conèixer l'obliqüitat primitiva dels caràcters utilitzats pels atributs, a fi i efecte de conèixer l'aixafament que han sofert i contrarestar-lo (funció **REST-OBLIC**), hauran de seguir un altra via, atès que la reordenació dels components alliberats és tan radical: una mena de joc de l'oca que tindrà per objecte rastrejar les dues fonts de dades (la definició del bloc i el resultant d'explosionar-ne una inserció) i anar saltant d'atribut en atribut; com que el reagrupament produït a la segona font no alterarà la posició relativa d'aquests components, si alternem les jugades en cadascuna de les fonts l'atribut resultant serà sempre el mateix. La funció **ATRIBS** s'ocuparà d'això, amb dos arguments que variaran segons la font des d'on s'hi accedeixi (**(ATRIBS E* ())** des de la definició del bloc i **(ATRIBS E T)** des del producte de la descomposició). **C:DESCOMPOK**, pel que fa al processat dels atributs típics l'hem subdividida en dos tractaments específics, segons que la inserció de **BLOC*** sigui obliqua o ortogonal:

```
; fragment de VERSIÓ 21++
(defun COMPR-OBLIC ()
  (not (= (cdr (assoc 41 LE)) (cdr (assoc 42 LE)) (cdr (assoc 43 LE)))))

(defun ATRIBS (E SS)
  (while (and E (or (not (setq LE (entget E) EZ (= (cdr (assoc 0 LE)) "ATTDEF")))
                    (= (logand (cdr (assoc 70 LE)) 2) 2)))
         (if SS
             (progn
               (if (and EZ (> EY 0) (wcmatch (cdr (assoc 2 LE)) "WWAA`-[12]"))
                   (progn
                     (entdel E)
                     (setq EY (1- EY))))
               (setq K (1+ K) E (ssname SA K)))
             (setq E (entnext E))))
  E)
```

```
(defun C:DESCOMPOK (/ Q0 ECO BLOC E LE E* SA A K N NEXTE EX EY EZ COMPROBLIC)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D (si hay atributos ")
  (prompt "con caracteres")
  (prompt "\noblicuos o no situados en su plano base, la orden DESCOMP no lo ")
  (prompt "resolverá bien).")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC (cdr (assoc 2 LE))
              BLOC (if (wcmatch BLOC "ATRIBATIP_#*")
                       (progn
                         (setq NEXTE E)
                         (while (or (/= (cdr (assoc 0 (setq E (ENTPRECEDING)
                                                              LE (entget E))))
                                        "INSERT")
                                    (if (wcmatch (cdr (assoc 2 LE))
                                                 "ATRIBATIP_#*")
                                        (setq NEXTE E))))
                         (cdr (assoc 2 LE)))
                       BLOC)
              ECO (getvar "CMDECHO"))
        (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
      ;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
        (setvar "CMDECHO" 0)
        (setq A (entnext E) K (= (last (trans (cdr (assoc 10 LE)) E 0)) 0))
        (command "DESCOMP" E ())
        (if (setq SA (ssget "P"))
            (progn
              (setq NEXTE (if NEXTE NEXTE (entnext E))
                    E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
                    COMPROBLIC (COMPR-OBLIC))
              (if (wcmatch BLOC (strcat "*_" N ",*_" N N))
                  (progn
                    (setq EZ (= (cdr (assoc 0 (entget (ssname SA 0)))) "INSERT")
                          K (if EZ 0 -1))
                    (while (setq K (1+ K) E (ssname SA K)
                                 E* (entnext E*))
                      (if (and (= K (if EZ 1 0))
                               (= (cdr (assoc 2 (entget E))) "NO-BLOQUEOK"))
                          (entdel E)
                          (if COMPROBLIC (REST-OBLIC)))))
                  (progn
                    (setq A (not (or (not A) (equal A NEXTE) COMPROBLIC
                                     (> (atoi (substr (getvar "ACADVER") 1 2)) 15)
                                     (and K (equal (cdr (assoc 210 LE))
                                                   '(0 0 1))))))
                          K 0 E (ssname SA K))
                    (if (wcmatch BLOC "*_AMB_ATRIBSTIPS")
                        (setq EX T EY 2)
                        (setq EY 0 LE (entget E*)
                              EX (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                      (= (cdr (assoc 2 LE)) "BLOQUEOK"))
                              EX (if EX EX (if A (AVIS)))))
                    (while (setq E (if (= K 0) (ATRIBS E T) E) E* (ATRIBS E* ()))
                           (if (and EX (wcmatch (cdr (assoc 2 LE)) "*BLOQUEOK"))
                               (progn (entdel E) (setq EX ()))
                               (if COMPROBLIC (REST-OBLIC)))
                           (setq K (1+ K) E (ssname SA K)
                                 E* (entnext E*)))))))
        (setq K 0)
        (while (and (setq E NEXTE)
                    (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                    (wcmatch (setq COMPROBLIC (COMPR-OBLIC)
                                   BLOC (cdr (assoc 2 LE)))
                             (setq A (strcat "ATRIBATIP_" (itoa K) "_DE_*"))))
```

```
                (command "DESCOMP" E ())
                (setq NEXTE (entnext E)
                      E (ssname (ssget "P") 0)
                      E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
                      E* (if (wcmatch BLOC (strcat A "_" N)) (entnext E*) E*))
                (if COMPROBLIC (REST-OBLIC)))
          (command "DESHACER" "F")
          (setvar "CMDECHO" ECO))
        (prompt "\n\nEsto no es ninguna inserción de bloque."))
    (princ))
```

Com haureu vist, hem incorporat el condicionament a **COMPROBLIC** (com que el test s'ha d'aplicar al bloc principal i a cadascun dels portadors d'atributs atípics, l'externalitzem amb la funció **COMPR-OBLIC**), que en la versió anterior ens havíem deixat: l'aixafament de l'altura dels atributs editables oblics només s'esdevé quan hi ha allò que havíem anomenat impròpiament "canvi d'obliqüitat" i que hagués estat millor qualificar de "no uniformitat en l'aplicació dels factors d'escala" (circumstància que implica també els seus signes, i no només els valors absoluts). I aclarirem quatre coses que al lector atent no li hauran passat desapercebudes:

1- La substitució del nom de variable **BLOC*** per **BLOC**.

2- Immediatament abans d'explosionar **E**, l'assignació
   (setq A (entnext E) K (= (last (trans (cdr (assoc 10 LE)) E 0)) 0))

3- Quan la inserció és ortogonal (**(not (wcmatch BLOC (strcat "*_" N ",*_" N N "_*")))**), la reassignació
   (setq A (not (or (not A) (equal A NEXTE) COMPROBLIC
   (and K (equal (cdr (assoc 210 LE)) '(0 0 1))))) ...)

4- Quan la inserció ortogonal és simple (inserció ortogonal en què es prescindeix del tàndem **NLOC-1**/**NLOC-2**), l'afegitó d'una segona assignació de valor a **EX**:
   (setq ... EX ... EX (if EX EX (if A (AVIS))))

La primera només és un recurs per tenir accés a la rutina **AVIS** (fins ara reservada a **SEGR-ATRIBS**) també des de **C:DESCOMPOK**, sense necessitat d'introduir un argument. ¿I per què volem que aparegui el missatge "*ATENCIÓN: Como <nom_BLOC> se creó con la orden BLOQUE y no con BLOQUEOK, los atributos justificados por su punto Medio pueden salir movidos*", quan **BLOC** no ha estat creat amb **C:BLOQUEOK** (no és **EX**), té atributs editables (és **A** però no **(equal A NEXTE)**), i la inserció explosionada és simple (context en el codi), no s'ha fet sota un ***SCP*** d'eix ***Z*** paral·lel al del ***SCU*** (no és **(equal (cdr (assoc 210 LE)) '(0 0 1))**) o, si s'hi ha realitzat, no ha estat en un punt del pla ***XY*** del ***SCU*** (no és **(= (last (trans (cdr (assoc 10 LE)) E 0) 0))**) i s'ha escalat uniformement (no és **COMPROBLIC**)? Doncs perquè el descobriment que havia obligat a preparar versions com la 20+ i 21+ només revelava una part de la malastrugança que porten els atributs editables justificats sobre el punt ***Medio***: si pàgines enrera advertíem que la seva posició quedaba mal registrada en la taula de blocs, si **BLOC** havia estat creat sota un ***SCP*** diferent del ***SCU****niversal* (o, per ser més exactes, si ho havia estat sota un ***SCP*** no paral·lel al ***SCU*** o el punt base no coincidia amb l'origen d'aquest últim), ara hem d'afegir que, amb independència de les circumstàncies de la seva creació, si explosionem una inserció simple de **BLOC** en què el pla ***XY*** no coincideix amb el pla ***XY*** del ***SCU*** (és a dir, l'eix ***Z*** no és paral·lel a l'eix ***Z*** del ***SCU*** o el punt d'inserció **I** no té una coordenada universal $\mathbf{I_z = 0}$) i realitzada amb factors d'escala $\mathbf{E_x = E_y = E_z}$, aquests atributs quedaran desplaçats respecte a la posició que haurien d'ocupar*. Tal i com vam acabar fent en relació a la primera anomalia (a la VERSIÓ 20+ es va donar un primer pas en la via analítica, mirant de qualificar-la i quantificar-la per aplicar una acció de signe contrari, fins que a la VERSIÓ 21+ va semblar més oportú prevenir que curar,

---

* En un altre lloc del present treball es comenta que el seu origen es remunta al 1996. Els primers passos es van donar, doncs, amb la versió 13 d'AutoCAD. Quan anys més tard l'autor va decidir tornar-hi, s'hi va posar amb la versió 15 (ACAD 2000) i únicament a les acaballes es va passar a la 16 (2004), tot i que en la seva activitat docent portava tres anys utilitzant-la. Llavors va descobrir, per pura xamba, que almenys l'anomalia que ara comentem (que, com totes les demés, havía trobat treballant amb ACAD 2000) ja no es produïa amb ACAD 2004. Com que l'autor és de l'opinió que la trajectòria més recent d'Autodesk, treient al mercat cada any una "nova versió" del producte, és una operació de marketing que potser té com a única explicació plausible la lluita contra la pirateria (en un percentatge massa alt, les innovacions d'un any per l'altre ho són més d'embolcall que de contingut), l'exit de la qual només s'explica per la manca de criteri d'una bona part dels joves usuaris (papanatisme tecnoconsumista, en podríem dir, si fos políticament i filològicament correcte), no ha volgut suprimir el recurs a **AVIS** i s'ha estimat més de respectar l'eventualitat d'usuaris d'ACAD 2000 i de versions anteriors (convençut que n'hi ha i que demostren tenir força seny, si la seva indiferència als cants de sirena dominants es basa en una estimació objectiva de les seves necessitats i no en atavismes de resistència a tot canvi), afegint a les existents la condició **(not (or ... (> (atoi (substr (getvar "ACADVER") 1 2)) 15)) ...)**.

evitant amb **C:BLOQUEOK** les circumstàncies en què es produïa), davant d'aquest nou contratemps actuarem igual, i no caldrà preparar cap altre dispositiu perquè, en derivar-se de la mateixa causa (els maleïts atributs amb justificació ***M***), ja ens va bé la mateixa teràpia. Tot i que, si volem ser rigorosos, hauriem de matisar:

- La transcripció errònia d'aquests atributs (registrats com a components de bloc) a la base de dades del dibuix es devia a la concurrència de dues circumstàncies:
  A) ***SCP*** en el moment de crear el bloc i punt base, en relació al ***SCU***.
  B) Justificació ***M*** de l'atribut.
  Cadascuna de les variants VERSIÓ 21A+ i VERSIÓ 21B+ de **C:BLOQUEOK** n'atacava una.

- Ara, el posicionament incorrecte després de l'explosió (conseqüència també d'una transcripció errònia a la base de dades, però considerats els atributs objectes independents) depèn de tres circumstàncies, una de les quals és B i les altres:
  C) Escalat uniforme de la inserció.
  D) ***SCP*** en el moment d'inserir el bloc i punt base, en relació al ***SCU***.

  Així que, si volem aprofitar la feina feta, només **C:BLOQUEOK** de VERSIÓ 21B+ ens protegirà de la nova incidència: era sota aquest pressupòsit (que, si el primer component de **BLOC** era l'atribut de nom "BLOQUEOK", l'hi havíem posat amb aquesta versió de ***BLOQUEOK*** i en conseqüència no podia haver atributs amb justificació ***M***) que una de les condicions per treure l'avís era **(not EX)**. Afegim, en relació a l'aparició de **AVIS**, que no la condicionarem a l'existència efectiva d'atributs amb justificació ***M*** i que, per estalviar-nos una cerca exhaustiva, n'hi haurà prou si en trobem d'editables (al cap i a la fi, **AVIS** es limita a dir-nos que "*... los atributos justificados por su punto Medio pueden salir movidos*"; tampoc no ens assegura que n'hi hagi). Una alternativa seria atacar el problema per C, canviant en **INSERTOK** l'expressió

```
(if (and NORM-XY NORM-Z)
    (progn
      (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                    (if ATREQ (append WWAA-1 WW))))
      (if OO-4 (INSERT** BL OX OY (* OZ I) 0)))
    (INSERT*))
```

  per

```
(if (and NORM-XY NORM-Z)
    (progn
      (if OO-4
          (INSERT** BL OX OY (* OZ I) 0)
          (if (= OX OY (* OZ I)) (setq OZ (* OZ (1+ Q0)))))
      (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                    (if ATREQ (append WWAA-1 WW)))))
    (INSERT*))
```

  Aquesta possibilitat, a la qual ens hi referirem com a VERSIÓ 21C+, té però una pega, ultra falsejar encara que sigui mínimament la mètrica de la inserció (tot i que, havent substituït el mode de justificació ***M*** per ***MC***, no tindria sentit que ara ens passéssim d'escrupolosos): afecta les insercions simples realitzades amb ***INS3D*** però no les obtingudes amb ***INSERT***, quan ambdós tipus d'insercions podrien ser explosionades amb ***DESCOMPOK*** (i convé que ho puguin ser si, amb independència de l'eina d'inserció, el bloc va ser fabricat amb ***BLOQUEOK***).

Altres punts foscos del codi de **C:DESCOMPOK** (VERSIÓ 21++) s'han deixat pel final, perquè responen més a qüestions de conveniència (ordre sintàctic) que conceptuals. Així, el lector potser es preguntarà perquè les expressions

```
(setq A (not (or (not A) (equal A NEXTE) COMPROBLIC
                 (> (atoi (substr (getvar "ACADVER") 1 2)) 15)
                 (and K (equal (cdr (assoc 210 LE)) '(0 0 1))))) ...)
```

i sobretot

```
(setq A (entnext E) K (= (last (trans (cdr (assoc 10 LE)) E 0)) 0))
```

s'avaluen tan prematurament, quan només interessen en el cas d'inserció simple de blocs creats amb ***BLOQUE***, és a dir, quan en l'última línia apareix la variable **A**:

```
(if (wcmatch BLOC (strcat "*_" N ",*_" N N))
    (progn ...)
    (progn ... (if (wcmatch BLOC "*_AMB_ATRIBSTIPS")
                   (setq ...)
                   (setq ... EX (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                     (= (cdr (assoc 2 LE)) "BLOQUEOK"))
                         EX (if EX EX (if A (AVIS)))))))
```

En la primera no hi ha més justificació que la conveniència de reutilitzar **K** com a contador, i en la segona és la necessitat d'efectuar la conversió de coordenades des del sistema propi de la inserció al ***SCU***, mitjançant la funció **trans**, allò que ens obliga a fer-ho abans d'explosionar aquesta inserció (després, l'identificador **(cdr (assoc 10 LE)** ja no apuntarà enlloc). Una altra cosa és si queda prou clar que la primera d'aquestes expressions respon als criteris enunciats més amunt en llenguatge descriptiu, i potser hi ajudarà mostrar les equivalències següents:

```
(setq A (and A
             (not (equal A NEXTE))
             (not COMPROBLIC)
             (<= (atoi (substr (getvar "ACADVER") 1 2)) 15)
             (or (not K) (not (equal (cdr (assoc 210 LE)) '(1 0 0))))) ...)
```

equival a

```
(setq A (and (not (not A))
             (not (equal A NEXTE))
             (not COMPROBLIC)
             (<= (atoi (substr (getvar "ACADVER") 1 2)) 15)
             (not (and K (equal (cdr (assoc 210 LE)) '(1 0 0))))) ...)
```

que, al seu torn, equival a l'expressió proposada

```
(setq A (not (or (not A)
                 (equal A NEXTE)
                 COMPROBLIC
                 (> (atoi (substr (getvar "ACADVER") 1 2)) 15)
                 (and K (equal (cdr (assoc 210 LE)) '(0 0 1))))) ...)
```

Abans d'abandonar l'accidentada vida dels atributs editables amb justificació ***M***, recordarem que la problemàtica comentada únicament afectarà blocs amb components tan conflictius quan tots els atributs (aquests i els altres) siguin típics: només llavors, l'aplicació de ***INS2D***/***INS3D*** a **BLOC** en les condicions **(and NORM-XY NORM-Z)** donarà lloc a insercions simples, i això si abans no hem fet cap inserció obliqua. Hem volgut acabar amb aquesta observació per tranquil·litzar aquells lectors que s'hagin quedat amb certa recança pel fet que ens haguem limitat a les fallades que podien esdevenir-se a ***DESCOMPOK***. Perquè algú podia pensar: ¿i si les insercions són obliqües, que no fem una primera inserció de **BLOC-2** amb $\mathbf{E_x = E_y = E_z}$ i després l'explosionem per crear el primer derivat genèric parcial **BLOC-2***?; ¿que no pot ser també que la inserció del bloc portador d'un atribut atípic, en les condicions **(not NORM-XY)**, es faci amb $\mathbf{E_x = E_y = E_z = 1}$ abans de ser explosionat?; ¿que potser no pot passar allò mateix que anunciàvem a **C:DESCOMPOK** en situacions idèntiques?. No, perquè les condicions no són idèntiques ni de bon tros: qui cregui que ho són és s'ha oblidat que, de la VERSIÓ 15+ ençà, els atributs justificats sobre el punt ***M*** han estat substituïts per atributs justificats sobre el punt ***C***, amb la posició d'aquest darrer calculada a **LINIA-BASE**.

També convé aclarir per què les crides a **ATRIBS** han quedat en la forma

```
(while (setq E (if (= K 0) (ATRIBS E T) E)
               E* (ATRIBS E* ())) ...)
```

perquè, si preteníem simplificar el codi, n'hi havia prou a fer

```
(while (setq E (ATRIBS E T)
               E* (ATRIBS E* ())) ...)
```

Ara bé: tret del cas d'inserció ortogonal amb escalat uniforme, sabem que un cop **(ATRIBS E T)** ens hagi posat en contacte amb el primer atribut típic el seguirà la resta sense solució de continuïtat, raó per la qual estalviarem noves (i estèrils) avaluacions d'aquesta expressió; i en el cas d'escalat uniforme ja no importa que amb aquesta simplificació pugui trencar-se la correspondència entre **E** i **E***, perquè sense **COMPROBLIC** la funció **REST-OBLIC** tampoc no s'executarà.

Volem cridar l'atenció del lector sobre el fet que, d'acord amb tot el que hem anat comentant, el perfil dels **BLOC***s destinats a rebre el primer tractament s'ha quedat reduït a **(wcmatch BLOC (strcat "*_" N ",*_" N N))**, perquè el supòsit **NLOC-1** (de perfil **(wcmatch BLOC "*_AMB_ATRIBSTIPS")**) passa a rebre el segon tractament, juntament amb les insercions simples del **BLOC** original. Tanmateix, en relació a aquestes últimes, té una introducció diferenciada de cara al seu accés a **ATRIBS**, accés que encara podríem haver singularitzat més, atès que és l'únic cas en què la definició del bloc explosionat i el conjunt d'objectes resultants de l'explosió tenen la mateixa estructura: de primer, els components que no són atributs P o N (editables), i tot seguit els que ho són; dintre de cada grup, a més, l'ordenació és idèntica. Que això es produïa a la segona col·lecció de dades (fora del cas en què $\mathbf{E_x = E_y = E_z}$) ho hem comentat abans, però ¿per què passa també a la primera?

Molt senzill: perquè **SEGR-ATRIBS** copia a la llista **OO-1** els components del **BLOC** original que no són atributs editables, i a la **OO-2** els atributs editables típics (la informació sobre els atípics queda repartida entre les llistes **OO-3** i **OO-4**); **NLOC-1** es forma amb el contingut de **OO-1** seguit del contingut de **OO-2**, tant si ho fem amb **MNLOC** perquè la primera inserció ortogonal és anterior a qualsevol altra d'obliqua (hi ha d'haver atributs atípics), com si ho fem amb **B->N** perquè **BLOC-1** i **BLOC-2** ja existien. Així, ens podríem treure de la màniga una nova variable **EW** per identificar aquest perfil i limitar l'accés a **ATRIBS** a només la primera iteració de **while** (localitzat el primer atribut típic, els altres ja vindran seguits):

```
        ..........................................
        (if (setq SA (ssget "P"))
            (progn
              (setq NEXTE (if NEXTE NEXTE (entnext E))
                    E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
                    COMPROBLIC (COMPR-OBLIC))
              (if (wcmatch BLOC (strcat "*_" N ",*_" N N "_*"))
                  (progn ...)
                  (progn
                    (if (setq A (not (or (equal A NEXTE) COMPROBLIC
                                         (and K (equal (cdr (assoc 210 LE))
                                                       '(0 0 1)))))
                                K 0 E (ssname SA K)
                                EW (wcmatch BLOC "*_AMB_ATRIBSTIPS"))
                        (setq EX T EY 2)
                        (setq EY 0 LE (entget E*)
                              EX (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                      (= (cdr (assoc 2 LE)) "BLOQUEOK"))
                              EX (if EX EX (if A (AVIS)))))
                    (while (if (= K 0)
                               (setq E (ATRIBS E T) E* (ATRIBS E* ()))
                               (setq E* (if EW E* (ATRIBS E* ()))))
                           (if (and EX (wcmatch (cdr (assoc 2 LE)) "*BLOQUEOK"))
                               (progn (entdel E) (setq EX ()))
                               (if COMPROBLIC (REST-OBLIC)))
                           (setq K (1+ K) E (ssname SA K)
                                 E* (entnext E*)))))))
                           ..................................................
```

El funcionament seria impecable, tot i haver passat per alt els canvis que podrien esdevenir-se dins del primer grup de components: únicament quan $\mathbf{E_x = E_y = E_z}$ queda garantida la correspondència entre elements; altrament, podem tenir reordenacions (els atributs Constants passaran a situar-se a la cua del grup, just abans del segon grup format per la resta d'atributs típics), i fins i tot un nombre diferent de components (l'explosió es propagarà als objectes múltiples, com polilínies o blocs inserits). Però convé preguntar-se si no estaríem embolicant massa la troca, perquè la qüestió és si val la pena de mantenir dos tractaments diferenciats, quan de fet el recurs a **ATRIBS** s'adapta perfectament a qualsevol inserció explosionada, sigui obliqua o ortogonal, perquè tant si els atributs típics P i N es desplacen cap enrera (és a dir, s'agrupen formant un conjunt compacte) com si no, guarden la posició relativa entre ells. És segur que amb la diferenciació evitaríem inútils accessos a aquesta funció, però una major velocitat de processat únicament seria significativa en blocs amb una molt elevada quantitat de components (i en concret, d'atributs editables), raó per la qual optem per conservar la feina feta sota el nom de VERSIÓ 21A++ però suggerim simplificar el codi per oferir una alternativa més compacta sota el nom de VERSIÓ 21B++ (no busqueu cap vincle entre aquests noms i les anomenades versions 21A+ i 21B+, que es referien a la funció **C:BLOQUEOK** i de les quals hem adoptat la segona, perquè només pensàvem en l'economia de lletres):

```
; fragment de VERSIÓ 21B++
(defun C:DESCOMPOK (/ Q0 ECO BLOC E LE E* SA A K N NEXTE EX EY EZ COMPROBLIC)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D (si hay atributos ")
  (prompt "con caracteres")
  (prompt "\noblicuos o no situados en su plano base, la orden DESCOMP no lo ")
  (prompt "resolverá bien).")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
```

```
(if (= (cdr (assoc 0 LE)) "INSERT")
    (progn
      (setq BLOC (cdr (assoc 2 LE))
            BLOC (if (wcmatch BLOC "ATRIBATIP_#*")
                     (progn
                       (setq NEXTE E)
                       (while (or (/= (cdr (assoc 0 (setq E (ENTPRECEDING)
                                                          LE (entget E))))
                                      "INSERT")
                                  (if (wcmatch (cdr (assoc 2 LE))
                                               "ATRIBATIP_#*")
                                      (setq NEXTE E))))
                       (cdr (assoc 2 LE)))
                     BLOC)
            ECO (getvar "CMDECHO"))
      (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
    ;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
      (setvar "CMDECHO" 0)
      (setq A (entnext E) K (= (last (trans (cdr (assoc 10 LE)) E 0)) 0))
      (command "DESCOMP" E ())
      (if (setq SA (ssget "P"))
          (progn
            (setq NEXTE (if NEXTE NEXTE (entnext E))
                  COMPROBLIC (COMPR-OBLIC)
                  A (not (or (not A) (equal A NEXTE) COMPROBLIC
                             (> (atoi (substr (getvar "ACADVER") 1 2)) 15)
                             (and K (equal (cdr (assoc 210 LE)) '(0 0 1)))))
                  K 0 E (ssname SA K)
                  E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
            (if (wcmatch BLOC (strcat "*_" N ",*_" N N ",*_AMB_ATRIBSTIPS"))
                (setq EX T EY 2)
                (setq EY 0 LE (entget E*)
                      EX (and (= (cdr (assoc 0 LE)) "ATTDEF")
                              (= (cdr (assoc 2 LE)) "BLOQUEOK"))
                      EX (if EX EX (if A (AVIS)))))
            (while (setq E (if (= K 0) (ATRIBS E T) E) E* (ATRIBS E* ()))
                   (if (and EX (wcmatch (cdr (assoc 2 LE)) "*BLOQUEOK"))
                       (progn (entdel E) (setq EX ()))
                       (if COMPROBLIC (REST-OBLIC)))
                   (setq K (1+ K) E (ssname SA K) E* (entnext E*)))))
      (setq K 0)
      (while (and (setq E NEXTE)
                  (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                  (wcmatch (setq COMPROBLIC (COMPR-OBLIC)
                                 BLOC (cdr (assoc 2 LE)))
                           (setq A (strcat "ATRIBATIP_" (itoa K) "_DE_*"))))
       (command "DESCOMP" E ())
       (setq NEXTE (entnext E)
             E (ssname (ssget "P") 0)
             E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
             E* (if (wcmatch BLOC (strcat A "_" N)) (entnext E*) E*))
       (if COMPROBLIC (REST-OBLIC)))
      (command "DESHACER" "F")
      (setvar "CMDECHO" ECO))
    (prompt "\n\nEsto no es ninguna inserción de bloque."))
(princ))
```

Abans de donar per acabat aquest capítol, ens creiem en l'obligació de revisar les versions 15+ i 16+ de **C:DESCOMPOK**, en els dos capítols precedents, a la llum de les descobertes realitzades. En ambdós casos, n'oferirem les variants "analítica" (15A++ i 16A++) i "compacta" (15B++ i 16A++), perquè l'usuari pugui triar una o altra en funció de les seves necessitats.

La revisió de VERSIÓ 15+ implicarà afegir-hi la funció auxiliar **ATRIBS**, comuna a les dues variants:

```
; fragment de VERSIÓ 15++
 (defun ATRIBS (E SS)
  (while (and E (or (/= (cdr (assoc 0 (setq LE (entget E)))) "ATTDEF")
                    (= (logand (cdr (assoc 70 LE)) 2) 2)))
         (if SS
             (setq K (1+ K) E (ssname SA K))
             (setq E (entnext E))))
  E)
```

A la variant d'estratègia diversificada (codi llarg), teòricament més eficient, hi afegirem a més la funció **REST-OBLIC**:

```
; fragment de VERSIÓ 15A++
(defun REST-OBLIC ()
  (setq A (cos (cdr (assoc 51 (entget E*)))))
  (if (not (equal A 1 Q0))
      (progn
        (setq LE (entget E)
              LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A)) (assoc 40 LE) LE)
              LE (subst (cons 41 (* (cdr (assoc 41 LE)) A)) (assoc 41 LE) LE))
        (entmod LE))))

 (defun C:DESCOMPOK (/ Q0 ECO BLOC* E E* LE SA A N)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INSERTOK, sin deformación de ")
  (prompt "sus atributos")
  (prompt "\n(si se definieron con caracteres oblicuos, usando DESCOMP se ")
  (prompt "reduce la altura):")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              ECO (getvar "CMDECHO"))
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (setvar "CMDECHO" ECO)
        (if (and (setq SA (ssget "P"))
                 (progn (setq K -1 E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))
                        (/= (cdr (assoc 41 LE)) (cdr (assoc 42 LE)))))
            (progn
              (if (not *EDITATRIBS-P*)
                  (while (wcmatch (cdr (assoc 0 (entget (entnext E*))))
                                  "ATTRIB,SEQEND")
                         (setq E* (entnext E*))))
              (if (wcmatch BLOC* (strcat "*_" N))
                  (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                         (REST-OBLIC))
                  (while (setq E (ATRIBS E T) E* (ATRIBS E* ()))
                         (REST-OBLIC)
                         (setq K (1+ K) E (ssname SA K)
                               E* (entnext E*)))))))
      (prompt "\n\nEsto no es ninguna inserción de bloque.")))
  (princ))
```

A la variant d'estratègia unificada (codi curt) no cal individualitzar cap funció **REST-OBLIC** perquè, en no haver-hi diversitat d'accessos, el seu contingut quedarà integrat directament a **C:DESCOMPOK**:

```
; fragment de VERSIÓ 15B++
 (defun C:DESCOMPOK (/ Q0 ECO BLOC* E E* LE SA A N)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INSERTOK, sin deformación de ")
  (prompt "sus atributos")
  (prompt "\n(si se definieron con caracteres oblicuos, usando DESCOMP se ")
  (prompt "reduce la altura):")
```

```
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              ECO (getvar "CMDECHO"))
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (setvar "CMDECHO" ECO)
        (if (and (setq SA (ssget "P"))
                 (progn (setq K -1 E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))
                        (/= (cdr (assoc 41 LE)) (cdr (assoc 42 LE)))))
            (progn
              (if (not *EDITATRIBS-P*)
                  (while (wcmatch (cdr (assoc 0 (entget (entnext E*))))
                                  "ATTRIB,SEQEND")
                         (setq E* (entnext E*))))
              (while (setq E (ATRIBS E T) E* (ATRIBS E* ()))
                     (setq A (cos (cdr (assoc 51 (entget E*)))))
                     (if (not (equal A 1 Q0))
                         (progn
                           (setq LE (entget E)
                                 LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A))
                                           (assoc 40 LE) LE)
                                 LE (subst (cons 41 (* (cdr (assoc 41 LE)) A))
                                           (assoc 41 LE) LE))
                           (entmod LE)))
                     (setq K (1+ K) E (ssname SA K) E* (entnext E*))))))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

El mateix passa a la revisió de VERSIÓ 16+, on la funció auxiliar **ATRIBS** (tornem a reproduir-la per evitar equívocs, malgrat ser igual a la de VERSIÓ 15+) és comuna a les dues variants:

```
; fragment de VERSIÓ 16++
 (defun ATRIBS (E SS)
  (while (and E (or (/= (cdr (assoc 0 (setq LE (entget E)))) "ATTDEF")
                    (= (logand (cdr (assoc 70 LE)) 2) 2)))
         (if SS
             (setq K (1+ K) E (ssname SA K))
             (setq E (entnext E))))
  E)
```

La variant d'estratègia diversificada (codi llarg), teòricament més eficient, juga amb **REST-OBLIC**, funció que no reproduïm perquè ja estava definida en VERSIÓ 16+:

```
; fragment de VERSIÓ 16A++
(defun REST-OBLIC ()
  (setq A (cos (cdr (assoc 51 (entget E*)))))
  (if (not (equal A 1 Q0))
      (progn
        (setq LE (entget E)
              LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A))
                        (assoc 40 LE) LE)
              LE (subst (cons 41 (* (cdr (assoc 41 LE)) A))
                        (assoc 41 LE) LE))
        (entmod LE))))

(defun C:DESCOMPOK (/ Q0 ECO BLOC* E LE E* SA A K N)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D, sin deformación de ")
  (prompt "sus atributos")
  (prompt "\n(si se definieron con caracteres oblicuos, usando DESCOMP ")
  (prompt "reducirían su altura):")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
```

```
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              ECO (getvar "CMDECHO"))
        (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
      ;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (setvar "CMDECHO" ECO)
        (if (and (setq SA (ssget "P"))
                 (progn (setq K -1 E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))
                        (not (= (cdr (assoc 41 LE)) (cdr (assoc 42 LE))
                                (cdr (assoc 43 LE))))))
            (progn
              (if (and (not *EDITATRIBS-P*) (wcmatch BLOC* (strcat "*_" N)))
                  (while (wcmatch (cdr (assoc 0 (entget (entnext E*))))
                                  "ATTRIB,SEQEND")
                         (setq E* (entnext E*))))
              (if (wcmatch BLOC (strcat "*_" N ",*_" N N))
                  (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                         (REST-OBLIC))
                  (while (setq E (ATRIBS E T) E* (ATRIBS E* ()))
                         (REST-OBLIC)
                         (setq K (1+ K) E (ssname SA K)
                               E* (entnext E*)))))))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

Com en la precedent i a diferència de VERSIÓ 15B++, en la variant d'estratègia unificada (codi curt) seguim utilitzant **REST-OBLIC**, funció que no reproduïm perquè ja estava definida a VERSIÓ 16+ (des de **C:DESCOMPOK** només hi ha un accés, però en tenim un altre a **INSPARCIAL**):

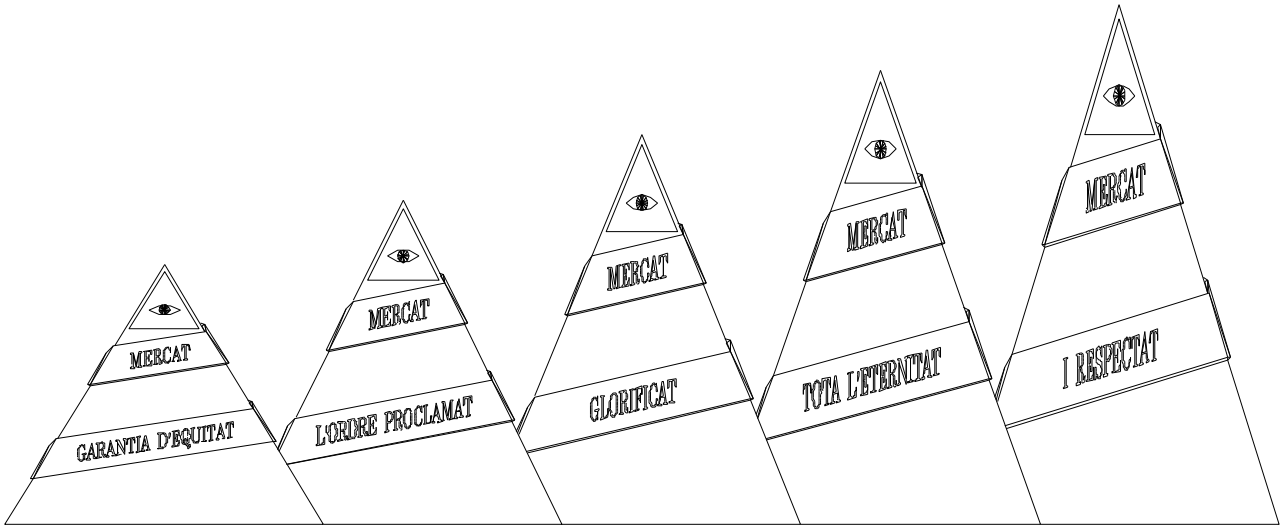
```
; fragment de VERSIÓ 16B++
(defun C:DESCOMPOK (/ Q0 ECO BLOC* E LE E* SA A K N)
  (setq Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D, sin deformación de ")
  (prompt "sus atributos")
  (prompt "\n(si se definieron con caracteres oblicuos, usando DESCOMP ")
  (prompt "reducirían su altura):")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC* (cdr (assoc 2 LE))
              ECO (getvar "CMDECHO"))
        (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
      ;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
        (setvar "CMDECHO" 0)
        (command "DESCOMP" E ())
        (setvar "CMDECHO" ECO)
        (if (and (setq SA (ssget "P"))
                 (progn (setq K -1 E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC*))))
                        (not (= (cdr (assoc 41 LE)) (cdr (assoc 42 LE))
                                (cdr (assoc 43 LE))))))
            (progn
              (if (and (not *EDITATRIBS-P*) (wcmatch BLOC* (strcat "*_" N)))
                  (while (wcmatch (cdr (assoc 0 (entget (entnext E*))))
                                  "ATTRIB,SEQEND")
                         (setq E* (entnext E*))))
              (while (setq E (ATRIBS E T)
                           E* (ATRIBS E* ()))
                     (REST-OBLIC)
                     (setq K (1+ K) E (ssname SA K) E* (entnext E*))))))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

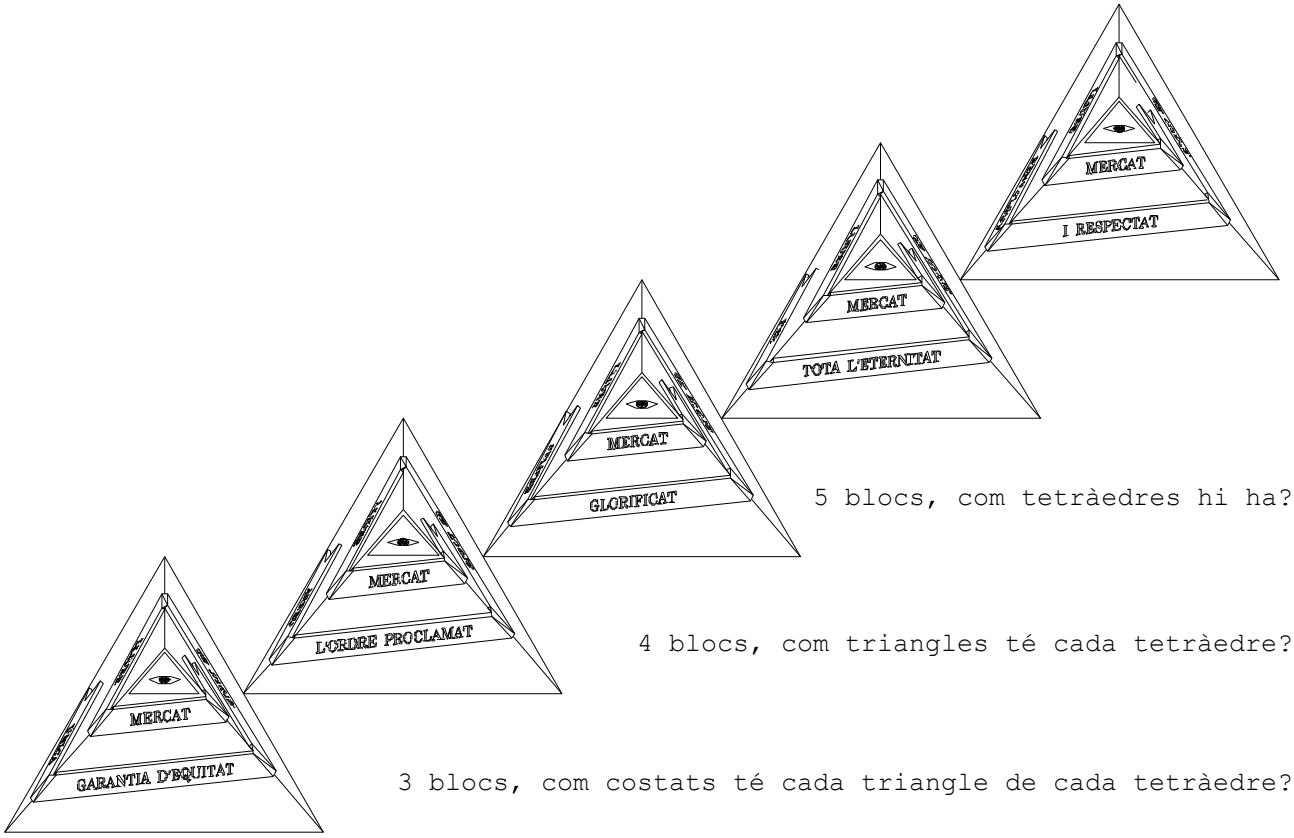
Ja hem acabat amb **C:DESCOMPOK**, tot i que, posats a revisar les versions anteriors, no tindria sentit que ens oblidéssim de la primera. En efecte, des de VERSIÓ 1 la bola de neu no ha parat de fer-se més i més grossa, per l'assumpció progressiva de fites més ambicioses (incorporació d'atributs típics, pas de 2D a 3D, incorporació d'atributs atípics) o per a la superació dels conflictes que s'anàven presentant. I, si amb una bona dosi d'optimisme creiem que les hores de feina enterrades en aquesta empresa han de servir per alguna cosa però volem ser sincers, haurem de reconèixer que, des del punt de vista de la pràctica corrent d'AutoCAD, l'ambit d'aplicació on pot tenir més fortuna és el de blocs 2D sense atributs, evidència que no resta validesa a l'afany d'haver volgut explorar els límits del sistema amb independència d'una demanda efectiva des del sector d'usuaris. I també des d'una perspectiva pràctica cal admetre que, a qui vulgui limitar-se a l'àmbit 2D, li fem un flac favor posant a la seva disposició una baluerna de la qual només aprofitarà ***INS2D***, que aplicat a blocs 2D sense atributs sols recull una petita part del codi. És pensant en aquest perfil d'usuari que tindria sentit recuperar la versió amb què culminàvem el capítol NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, per bé que actualitzada amb les aportacions pertinents dels restants: ENCAIX 2D DE BLOCS AMB ATRIBUTS (on simplificàvem el panorama adoptant sempre genèrics sense simetria respecte al bloc original, decisió transcendent que ens duia a incorporar a VERSIÓ el primer +); ENCAIX 3D DE BLOCS AMB ATRIBUTS i el present ENCAIX 3D DE BLOCS AMB ATRIBUTS SITUATS LLIUREMENT (amb el dispositiu ***DESHACER Inicio/Fin***, introduït a la VERSIÓ 16+, bandejat provisionalment a la VERSIÓ 17+ i recuperat a la VERSIÓ 19+); fins i tot ens permetrem la llicència de viatjar en el temps i avançar algunes de les correccions relatives a compatibilitat de toleràncies en el càlcul que adoptem en el capítol ENCAIX 3D ...: OPTIMITZACIÓ DE L'ESTRUCTURA DE BLOCS que vindrà tot seguit (ho fem per no "contaminar" amb afegitons extemporanis el capítol final i perquè semblava que on som ara era el lloc més adient, perquè ja no en ve d'una).

Heus aquí, millorada, la primera VERSIÓ 1:

```
; VERSIÓ 1++

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX N C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "CLAYER" "0" "INSERT" BLOC () "CLAYER" CAPA)
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun ANG (S C / A B)
  (setq A (/ (atan S C) PI/2 Q0) B (/ 1 Q0)
        A (if (< (- B A) 1) (1- B) (if (< A 1) 1 A))
        B (fix A) A (itoa (if (> (- A B) 0.5) (1+ B) B))
        B (itoa (1- (fix (/ 1 Q0)))))
  (repeat (- (strlen B) (strlen A)) (setq A (strcat "0" A)))
  A)

(defun INSERT* (/ X* X** Y** OX* OX** OY** BLOC*)
  (setq W (if (< (expt W 2) (+ (expt OX 2) (expt OY 2))) (angle B X) (angle X B))
        B (mapcar '/ (mapcar '+ B X) '(2 2))
        W (angle O (polar B W (distance O B)))
```

```
      X* (inters X (polar X W 1) O B ())
      X** (inters O (polar O (+ W PI/2) 1) X X* ())
      Y** (inters Y (polar Y W 1) O X** ())
      OX* (distance O X*) OX** (distance O X**) OY** (distance O Y**)
      BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))
      BLOC* (strcat BLOC "_" (ANG OY** OX**))
      K (tblsearch "BLOCK" BLOC*))
  (if (not K)
      (command "CLAYER" "0"
               "CECOLOR" "PORBLOQUE"
               "CELTYPE" "PORBLOQUE"
               "CELWEIGHT" -2
               "INSERT" BLOC O (setq K (/ OX* OX**)) K X*
               "SCP" "Z" (setq K (trans O 1 0) O O) X**
               "BLOQUE" BLOC* (trans K 0 1) "LT" ""
               "SCP" "PR"
               "CELWEIGHT" GLIN
               "CELTYPE" TLIN
               "CECOLOR" COL
               "CLAYER" CAPA))
  (command "INSERT" BLOC* O OX** (* (distance Y** Y) I) X**))

(defun C:INSERTOK (/ Q0 PI/2 CAPA COL TLIN GLIN ECO OSN BLOC O X Y OX OY A B E LE
                     W I J K)
  (setq Q0 0.001
      PI/2 (/ PI 2)
      CAPA (getvar "CLAYER")
      COL (getvar "CECOLOR")
      TLIN (getvar "CELTYPE")
      GLIN (getvar "CELWEIGHT")
      ECO (getvar "CMDECHO")
      OSN (getvar "OSMODE")
      O (getvar "INSNAME")
      BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
      BLOC (if (= BLOC "")
               (if (> O "") O (prompt "\nNombre de bloque no válido."))
               (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                            (< (cdr (assoc 70 O)) 2))
                       (and (not O) (findfile (strcat BLOC ".dwg"))))
                   BLOC (alert (if O (REFX) (RUTES)))))
      BLOC (if BLOC (strcase BLOC) (exit))
      O (getpoint "\nPunto de inserción: "))
  (foreach P '("X" "Y")
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +" P
                                  " desde el punto de inserción: "))
            I (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC " <1>: "))
            W (angle O A) B (/ (distance O A) (if I I 1)))
      (set (read P) (polar O W B))
      (set (read (strcat "O" P)) B))
  (setq A (polar O (+ W PI/2) OY) B (polar O (- W PI/2) OY)
      W (distance X Y)
      I (if (< (distance A X) (distance B X)) -1 1)
      J (- (* (- (car X) (car O)) (- (cadr Y) (cadr O)))
           (* (- (cadr X) (cadr O)) (- (car Y) (car O))))
      K (* OX OY))
  (setvar "CMDECHO" 0)
  (command "DESHACER" "I"
           "OSMODE" 0)
  (if (> J (* K Q0))
      (if (equal (/ J K) 1 (- 1 (sin (+ PI/2 Q0))))
          (command "INSERT" BLOC O OX (* OY I) X)
          (INSERT*)))
  (command "INSNAME" BLOC
           "OSMODE" OSN
           "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))
```





5 blocs, com tetràedres hi ha?

4 blocs, com triangles té cada tetràedre?

3 blocs, com costats té cada triangle de cada tetràedre?

2 blocs, com extrems té cada costat de cada triangle de cada tetràedre?

No, cars germans en el Lliure Mercat: un únic bloc, global i indivisible com Ell.

ENCAIX 3D...: OPTIMITZACIÓ DE L'ESTRUCTURA DE BLOCS

A la INTRODUCCIÓ ja s'havia comentat que aquesta SEGONA PART reprèn uns treballs interromputs fa onze anys, la idea motriu dels quals fou la millor adequació del mètode gràfico anomenat de Ritz (determinació dels eixos d'una el·lipse a partir de dos diàmetres conjugats) a la mecànica d'inserció dels blocs d'AutoCAD, en relació a la resolució algebraica convencional basada en transformacions lineals. Amb uns resultats que, des de la perspectiva actual, calificaría de "manifestament millorables", en data 28-01-97 vaig contactar amb Autodesk Spain per proposar-los un tracte: aquella modesta aportació al desenvolupament d'aplicacions per al seu producte estrella a canvi de llicències, per tal de regularitzar la precària cobertura legal amb què tenia que impartir docencia en l'E.T.S. d'Enginyers de Telecomunicació de Barcelona. Com era de preveure, a la vista de la poc encertada política comercial d'aquesta delegació (és una opinió personal que no persegueix desacreditar ningú, perquè el problema pot ser que des de San Rafael –Califòrnia, USA- o Neuchatel –Neuchatel, CH- la tinguin lligada massa curt), no vaig tenir cap resposta, i la solució al problema de les llicències va venir d'una altra banda. I, com que amb la consecució d'aquests resultats (mínims, però resultats al cap i a la fi) el tema va deixar d'estimular-me i uns altres en van prendre el relleu, els treballs van quedar en hibernació fins que uns anys després vaig decidir recuperar-los en el marc d'una publicació de més abast que tingués de protagonista la inserció de blocs AutoCAD tractada des del vessant geomètric, força descuidat en relació a d'altres perspectives.

Aquest preàmbul té com a objectiu donar pas a confessions encara més personals, no pas guiades per afanys exhibicionistes sinó perquè el lector pugui entendre la raó d'un cert desgavell expositiu en relació a un discurs que hauria d'avançar més decididament cap a la meta a on se suposa que l'autor ja havia arribat (una cosa és comentar algún error amb finalitats didàctiques, i l'altra obligar el primer a passar el mateix calvari que el segon fent marrades innecessàries). El caràcter de moltes giragonses té més d'erràtic que no d'astuta maniobra d'inspiració leninista per poder avançar dos passos després d'haver-ne reculat un, i cal revelar-ne les causes prosaiques sense fals pudor: en molts casos simplement ha passat que, ja era prou feixuc desxifrar documents de feia anys (sovint el codi AutoLISP pelat, sense comentaris ni dibuixos, que d'entrada resultava tan hermètic com si hagués estat obra d'altri), com per detectar errors (de programació o de plantejament) o incoherències en relació a la nova línia argumental que anava quedant reflectida en el text present; en d'altres, el règim de dedicació a un treball que s'anat obrint pas per pura obstinació personal i sense cap estímul departamental (parlo de suport moral, perquè amb facilitats d'un altre tipus tampoc no hi comptava), aprofitant els buits deixats per les obligacions docents, intermitència en què sovint calia interrompre el treball just quan diversos interrogants havíen quedat oberts, de vegades per períodes llargs, amb les dificultats consegüents per tornar a connectar-hi; tot i que de vegades aquestes llargues pauses obligades han tingut el seu vessant positiu, en donar cabuda a reflexions que han servit per prendre consciència d'estar en un cul de sac, han permès la gestació d'idees alternatives i més eficients o han afavorit la reformulació de fites més ambiciosos en relació als primers objectius, tendència que pot ser perillosa si un no la sap controlar. En qualsevol cas, la memòria extremadament volàtil de l'autor, la seva impaciència i un elemental sentit de la realitat que li aconsellava no posposar la descripció del procés a la consecució dels últims objectius en matèria de programació, amb una irregular i defectuosa sincronia entre el progrés en aquesta matèria i la seva traducció narrativa, són els responsables d'aquesta línia argumental vacil·lant. D'alguna manera, aquest text de lectura indigesta fins i tot per al seu autor té més de quadern de bitàcora, concebut per anar-hi dibuixant la ruta recorreguda i poder-s'hi ubicar quan els arbres ja no deixaven veure el bosc, que de exposició sistemàtica orientada als demés o, menys encara, de manual d'ús. És per això que la trajectòria descrita recorda més una gràfica estadística, de signe globalment ascendent (confiem que sigui així) però amb les singladures en ziga-zaga, que a una corba contínua sense gairebé mínims locals.

El títol del capítol respon bàsicament a unes petites modificacions que portaran, però, importants conseqüències pel que fa a la simplificació de l'estructura de genèrics del sistema: quedaran recollides en la VERSIÓ 23++, última i definitiva.

Però abans volem depurar la VERSIÓ 21++ de tot un munt d'incoherències i qüestions menors (o no tant) pendents de resolució, aplegant totes aquestes accions prèvies en la que anomenarem VERSIÓ 22++.

La primera millora que presenta la nova VERSIÓ 22++, en relació a la VERSIÓ 21++, reflecteix molt bé una d'aquestes vacil·lacions, en la mesura que posa fi a una trajectòria més aviat erràtica pel que fa al tracte que hem anat dispensant als atributs verificables: a l'inici del capítol precedent (VERSIÓ 17+) decidíem de prescindir en els blocs substituts de la característica *Verificable* dels atributs típics i atípics, argumentant que ja prou que s'embolicaria la troca en incorporar aquests últims a ***INS2D***/***INS3D***; més tard (VERSIÓ 19+) decidíem de mantenir-la perquè ens adonàvem que la supressió donava lloc a una arbitrarietat de funcionament del tot inadmissible (atributs verificables de **BLOC** que es comportàven com a tals en unes ocasions –insercions ortogonals, en determinades circumstàncies- però no en d'altres –insercions obliqües, i també ortogonals en circumstàncies diferents-), tot i reconeixent "... l'extemporània repetició de l'avís *Verificar valores de atributos* sempre que **BLOC** tingui atributs d'aquesta mena, inconveniència que ara se'ns agrava perquè si són atípics es resoldran amb insercions addicionals ...". L'objectiu serà doncs evitar l'aparició en pantalla de l'irreductible anunci, quan el programa insereixi efectivament els genèrics **BLOC*** (**(not NORM-XY)** i **NORM-Z**), **BLOC**** (**(not NORM-Z)**) i sobretot els dels blocs de suport de cada atribut atípic, atès que l'usuari està convençut que l'assignació de valor ja s'havia realitzat, i el procediment consistirà a suprimir la característica *Verificable* dels atributs de **BLOC-2**, de **NLOC-1** i dels blocs portadors (cosa que així, sense més, pot semblar una regressió a la VERSIÓ 17+), però aquest cop "marcant" tots els afectats per la supressió, de manera que la sol·licitud prèvia de valor incorpori una confirmació en segona volta i que, tanmateix, a l'hora de les insercions finals ***INSERT*** només es necessiti un ínput per a cada atribut N.

El sistema de marcatge és tant rudimentari com modificar el nom identificador de l'atribut alhora que li arrabassem la característica *Verificable* (li empeltem el prefix "VRF*"), però ens hem d'assegurar que l'usuari mai no s'assabentarà de la manipulació. En les funcions **VATR**, **VAL-ATRIBS-1** i **SEGR-ATRIBS** (com que és llarga, només en reproduïm un fragment) hem subratllat els canvis implicats a la jugada:

```
(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "")
                                          (if (wcmatch N "VRF`**") (substr N 5) N)
                                          M)
                                 (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS-1 (LA / N M V)
  (if (= (logand ICVP 8) 0)
      (progn
        (if (and ATREQ (not INI))
            (setq INI (not (prompt "\nIndique valores de atributo"))))
        (VATR (wcmatch (cdr (assoc 2 LA)) "VRF`**") (cdr (assoc 2 LA))
              (cdr (assoc 3 LA)) (cdr (assoc 1 LA)))
        (setq LLAA (cons (list W N M V) LLAA)))))

(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 NL1EX NL2EX BL1EX BL2EX BLOK AT AT-C
                      NE E1 E2 EXT INI LLAA C-10 C-11 C-72 C-74 OO-3 OZ VZ FW)
  ..........................................................................
                        (setq NE (1+ NE)
                              C-72 (cdr (assoc 72 LE)) C-74 (cdr (assoc 74 LE))
                              BLOK (if (and (= BLOK 0) (= C-72 4))
                                       (progn
                                         (setq E1 (AJUSTA-E E1) E2 (AJUSTA-E E2)
                                               E1 (strcat " 0" E1) ...)
                                         (AVIS))
                                       BLOK)
                              LE (if (= (logand ICVP 4) 4)
                                     (subst (cons 2 (strcat "VRF*"
                                                            (cdr (assoc 2 LE))))
```

```
                                    (assoc 2 LE)
                                    (subst (cons 70 (setq ICVP
                                                          (- ICVP 4)))
                                           (assoc 70 LE)
                                           LE))
                             LE)
                      LE (if (= C-72 4)
                             (subst (cons 72 (setq C-72 1))
                                    (cons 72 4)
                                    (subst (cons 74 (setq C-74 2))
                                           (assoc 74 LE)
                                           LE))
                             LE)
                      ........................................
                      OZ (last (assoc 10 LE))
                      VZ (cdr (assoc 210 LE)))
                      ......................
```

Ja suposareu que, tant en **INSERT**** com en **INSERT***, caldrà prescindir del suplement **WW** que aportava precisament a ***INSERT*** els arguments "" de confirmació corresponents als atributs verificables. En la primera funció, podem eliminar la línia

```
                WW (if (and WWAA-1 (= (logand ICVP 4) 4)) '(""))
```

de l'assignació que obre cadascuna de les **OO-4** iteracions, i l'expressió

```
          (if NORM-XY
              (eval (append '(command "INSERT" BLOC O OX OY 0)
                            (if ATREQ (append WWAA-1 WW))))
```

ha de quedar reduïda a

```
          (if NORM-XY
              (eval (append '(command "INSERT" BLOC O OX OY 0)
                            (if ATREQ WWAA-1)))
```

igual que, en la segona, l'expressió

```
  (eval (append '(command "INSERT" BLOC* O "XYZ" OX** (setq J (distance Y Y**))
                          (setq K (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1)
                                     OX** I)) X**)
                (if ATREQ (append WWAA-1 WW))))
```

ha de quedar reduïda a

```
  (eval (append '(command "INSERT" BLOC* O "XYZ" OX** (setq J (distance Y Y**))
                          (setq K (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1)
                                     OX** I)) X**)
                (if ATREQ WWAA-1)))
```

Hauríem d'eliminar també de **VAL-ATRIBS-2** tota referència a **WW**? Ens reservarem el dret de rectificar més endavant, a la vista d'altres consideracions, però ara com ara no sembla que puguem ser tan radicals, perquè tal i com ho tenim muntat encara hi ha un cas en què necessitem aquesta llista per confirmar les assignacions fetes als atributs típics verificables: la inserció ortogonal d'un **BLOC** sense atributs atípics, executada mitjançant ***INS2D***/***INS3D*** abans d'haver realitzat cap inserció obliqua (o després, però havent fet una poda severa de la qual ni **BLOC-1**/**BLOC-2** ni **NLOC-1**/**NLOC-2** no n'hagin sortit sencers). En aquestes circumstàncies (casos **1**, **3**, **5**, **9**, **11**, **13**, **15**, **17**, **19**, **21**, **23**, **27** i **29** de la "taula de veritat" que construïem en el capítol precedent), si considerem el fragment següent del codi de **INSERTOK**

```
  ......................
  (SEGR-ATRIBS)
  (if (and NORM-XY NORM-Z)
      (progn
        (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                      (if ATREQ (append WWAA-1 WW))))
        (if OO-4 (INSERT** BL OX OY (* OZ I) 0)))
      (INSERT*))
  ..........................................
```

i recordem l'anàlisi exhaustiva de **BLOC** a **SEGR-ATRIBS**, d'on sortien les llistes **OO-1** i **OO-2**, la llista **WWAA-1** de valors assignats als atributs i també el text **E1**, però que no culminava en la creació del tàndem **NLOC-1**/**NLOC-2**, en no adequar-se el seu perfil **(and (not OO-4) NORM)** al filtre **(or OO-4 (not NORM))** que dóna accés a la funció **MNLOC** (**NORM** equival a **(and NORM-XY NORM-Z)**), veurem que la variable **BLOC** encara representa el **BLOC** original, nom que havíem traslladat a **BL** per protegir-lo de la reassignació amb el de **NLOC-1** practicada en els altres casos **NORM** (**7**, **25**, **31** i, amb atributs atípics, els que acabem d'esmentar). Això vol dir que els atributs

verificables mantenen aquesta característica i que cal preservar la llista **WW** de textos nuls per ampliar **WWAA-1** amb els arguments de confirmació; per contra, quan sota la variable **BLOC** s'amagui el nom de **NLOC-1** caldrà anul·lar **WW**, perquè estarem treballant amb atributs "VRF*..." ja despossetts de la característica *Verificable*. Tot plegat es podria solucionar afegint el codi subratllat al fragment de **INSERTOK**

```
  ......................
  (SEGR-ATRIBS)
  (if (and NORM-XY NORM-Z)
      (progn
        (if (/= BLOC BL) (setq WW ()))
        (eval (append '(command "INSERT" BLOC O "XYZ" OX OY (* OZ I) 0)
                      (if ATREQ (append WWAA-1 WW))))
        (if OO-4 (INSERT** BL OX OY (* OZ I) 0)))
      (progn
        (setq WW ())
        (INSERT*)))
  ..........................................
```

tot i que serà més senzill deixar-ho resolt a **SEGR-ATRIBS** (nou codi, subratllat):

```
(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 ... C-10 C-11 C-72 C-74 OO-3 OZ VZ FW)
  ..........................................................................
  (if (and (or NL2EX BL2EX) (not (XOR NL1EX NL2EX)) (not (XOR BL1EX BL2EX)))
      (progn
        ........................
        (INICOMMAND)
        (if (and (not NL1EX) NORM)
            (MNLOC)
            (if (not (or BL1EX NORM))
                (MBLOC)
                (if NORM (setq BLOC NLOC-1))))
        (setq WW ()))
      (progn
        (setq  E (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
               BLOK E OO-4 () NE -1 E1 "" E2 "")
        (while E
               ..................
               (setq E (entnext E)))
        (VAL-ATRIBS-2)
        (INICOMMAND)
        (if FW (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))
        (if (or OO-4 (not NORM))
            (progn
              (command "REGENT"
                       "CECOLOR" "PORCAPA"
                       "CELTYPE" "PORCAPA"
                       "CELWEIGHT" -1)
              ................................
              (if (or BL1EX OO-4) (REDEF-BLOC*S))
              (setq WW ())
              (setvar "CECOLOR" COL)
              (setvar "CELTYPE" TLIN)
              (setvar "CELWEIGHT" GLIN))))))
```

Allò que sí que podem fer és suprimir de la funció **BLOC->WWAA** la cinquena línia,

```
        WW (car E1) E1 (cdr E1)
```

corresponent a l'assignació múltiple inicial, i netejar mínimament **VAL-ATRIBS-2**, considerant que incloure el nombre d'atributs típics verificables (longitud de la llista **WW**) en el text **E1** que emmagatzemem a l'atribut "WWAA-1" de **BLOC-1** i **NLOC-1**, ja no té cap sentit: únicament necessitem **WW** quan inserim directament **BLOC**, cas en què aquest no té atributs atípics (**(not E1E2)**) i en què els tàndems **BLOC-1**/**BLOC-2** i **NLOC-1**/ **NLOC-2** (**(not EXT)**) ni existeixen ni de moment no ens fan cap falta, raó per la qual podem treure de **VAL-ATRIBS-2** l'expressió

```
  (if EXT
      (if E1E2 (repeat WW (setq W (cons "" W))))
      (setq E1 (strcat (itoa (length W)) E1)))
```

tot i que deixant l'última línia

```
  (setq WW W))
```

Però no ha acabat tot aquí: pensem que la manipulació dels atributs verificables, que tan zelosament hem ocultat a l'usuari, quedarà en evidència quan explosionem les insercions fetes amb ***INS2D***/***INS3D***; ja no sols perquè els components resultants estiguin adulterats en relació al seu estat quan van entrar a formar part de **BLOC** (en això, els atributs verificables que han deixat de ser-ho comparteixen aquest caràcter espuri amb els de justificació ***Medio*** i amb tots els demés no justificats sobre la línia base) sinó perquè el canvi de nom ho va proclamant als quatre vents (recordeu que, en descompondre un bloc, els seus atributs perden el valor assignat i es mostren mitjançant els identificadors, que en el cas dels verificables han rebut l'empelt del prefix "VRF*"). Doncs bé, igual que ***DESCOMPOK*** ens havia permès d'eliminar els atributs auxiliars "WWAA-1", "WWAA-2", "BLOQUEOK" i "NO-BLOQUEOK", a més de contrarestar l'aixafament transversal esdevingut sota certes condicions, actuacions que tenien en comú la supervisió del resultat de la descomposició, si més no en l'aspecte visual, ara li demanarem dues prestacions més: mantenint-nos en el nivell formal, treure als identificadors l'afegitó "VRF*"; i, ja posats a la detecció d'aquests atributs, donarem un pas més i els retornarem la característica *Verificable* que els havia estat arrabassada. Totes dues accions estaran a càrrec de la funció **REST-VERIF**, bessona de **REST-OBLIC** i amb qui col·laborarà en el marc de **C:DESCOMPOK**. El fet que a **REST-OBLIC** també s'hi accedeixi des de **INSPARCIAL** ens ha impedit de refondre-les en una única funció, i per no reiterar les invocacions dobles s'ha recorregut a la funció portadora **RESTAURA**. Veieu de moment **REST-OBLIC** i **RESTAURA**, que **C:DESCOMPOK** (de la qual mantindrem les variants A i B del capítol precedent) us l'oferirem d'aquí a no res, amb el conjunt del codi:

```
(defun REST-VERIF ()
  (setq A (cdr (assoc 2 (entget E*))))
  (if (wcmatch A "VRF`**")
      (progn
        (setq LE (entget E)
              LE (subst (cons 2 (substr A 5)) (cons 2 A) LE)
              LE (subst (cons 70 (+ (cdr (assoc 70 LE)) 4)) (assoc 70 LE) LE))
        (entmod LE))))

(defun RESTAURA () (REST-VERIF) (if COMPROBLIC (REST-OBLIC)))
```

Havent-nos molestat amb **REST-VERIF** a recuperar la composició original de **BLOC**, ara ens podríem preguntar perquè no fer el mateix pel que fa a la substitució dels atributs editables que inicialment tenien una justificació ***Medio*** i que, en adoptar en la VERSIÓ 21B+ la alternativa de **C:BLOQUEOK** que prescindia de **ALINEAR+**, havíem passat a justificació ***MC***. La raó és senzilla: l'usuari havia estat advertit de la substitució amb l'avís *ATENCIÓN: En los atributos editables seleccionados, el modo de justificación M se sustituye por MC* (calia informar-lo, perquè la manipulació afectava el bloc públic **BLOC**), mentre que de la desactivació de la característica *Verificable* no l'havíem informat (la manipulació sols afectava els substituts d'ús intern **BLOC-2**, **NLOC-1** i **NLOC-2**). De tota manera, si per alguna raó convingués que tots els atributs de justifició ***MC*** procedents de l'explosió passessin a ***M***, no hi ha cap problema: podríem definir una funció **REST-JUST-MC->M**, calcada de **REST-VERIF** i **REST-OBLIC**,

```
(defun REST-JUST-MC->M ()
  (setq A (entget E*))
  (if (and (= (cdr (assoc 72 B)) 1) (= (cdr (assoc 74 B)) 2))
      (entmod (subst (cons 72 4) (cons 72 1) (entget E)))))
```

i ampliar **RESTAURA** a

```
(defun RESTAURA () (REST-JUST-MC->M) (REST-VERIF) (if COMPROBLIC (REST-OBLIC)))
```

però cal adonar-nos que aquesta acció seria indiscriminada, perquè tot i que podem saber si **BLOC** va ser construït amb **C:BLOQUEOK** (perquè el bloc explosionat era un genèric **BLOC****, **BLOC*** o un substitut **NLOC-1**, i entre els components alliberats no s'ha localitzat cap atribut anomenat "NO-BLOQUEOK", o perquè era el mateix **BLOC** i hi hem trobat l'atribut "BLOQUEOK", totes elles situacions fàcilment detectables amb una mínima intervenció sobre el codi de **C:DESCOMPOK**), no queda constància de quins van ser els atributs manipulats (en quedaria si, per exemple, modifiquéssim l'identificador d'aquests atributs amb un prefix, com s'ha afegit el prefix "VRF*" al dels verificables, però tot plegat ja sembla excessiu) i quins ja havíen estat definits d'entrada amb justificació ***MC***. Tanmateix, pot semblar que restaurar els atributs inicialment no justificats sobre la línia base és abordable sense massa complicacions, amb alguna funció **REST-JUSTIF** similar a **REST-JUST-MC->M**, perquè, en haver estat **BLOC-2** i **NLOC-2** els blocs manipulats, només caldria comparar els seus atributs amb els homònims de **BLOC** (quan detectés diferències, **REST-JUST-MC->M** es

limitaria a copiar els codis 74 i 11, sense necessitat de calcular-ne les últimes posicions seguint un procés invers al de **LINIA-BASE**), però sempre ens quedaria el dubte de si inicialment **BLOC** tenia tots els atributs justificats des de la línia base (havent passat a **BLOC-2** i **NLOC-2** conservant aquesta justificació) i després va ser redefinit però sense que l'usuari de ***INS2D***/***INS3D*** traslladés a l'aplicació la nova definició de **BLOC** seguint el protocol establert (la destrucció de **BLOC-2** o de **NLOC-2**): a algú li pot semblar excessivament recargolat imaginar aquest supòsit (si més no, tan improbable com que un identificador d'atribut comenci amb "VRF*"); doncs qui opini així que incorpori a **RESTAURA** alguna funció **REST-JUST-MC->M** en la línia del perfil traçat, perquè aquí donarem per tancat el tema sense fer-ho.

Ens preocupa força més que l'objectiu que guiava aquesta millora de la VERSIÓ 21++ (evitar la repetició extemporània del missatge *Verificar valores de atributos*, que amb la presència d'atributs atípics verificables es podia multiplicar) no s'hagi acomplert al cent per cent. Perquè, després de tant d'enrenou, encara hi ha una situació en la qual *Verificar valores de atributos* sortirà en pantalla quan ja tot estigui dat i beneït: la mateixa que ens havia obligat a recuperar el contingut de **WW** per ampliar la llista **WWAA-1** d'assignacions de valor als atributs típics, just abans d'estendre la millora a **C:DESCOMPOK**. La inserció era ortogonal però, en no tenir **BLOC** atributs atípics ni estar condicionats per la preexistència del tàndem **BLOC-1**/**BLOC-2**, en lloc d'utilitzar un substitut **NLOC-1** inseríem directament **BLOC**, que mantenía la característica *Verificable* dels atributs, constituint-se així en el taló d'Aquil·les del nostre invent.

L'única manera de silenciar l'eco irreductible serà seguir una estratègia oposada a la que havíem adoptat per adequar el contingut de la llista **WW** al comportament dels atributs del bloc inserit, pel que fa a la característica esmentada: si les funcions **VAL-ATRIBS-2** i **BLOC->WWAA** es molestaven a crear aquesta llista de textos nuls (de la informació sobre la seva llargada, que abans incorporàvem al text **E1** emmagatzemat a l'atribut constant i invisible "WWAA-1" de **BLOC-1** i/o **NLOC-1**, ja n'havíem prescindit), el contingut de la qual només s'aprofitava quan l'objecte de la inserció era el **BLOC** original, y en cas contrari havíem d'anul·lar-la, el que ara proposem és l'ús generalitzat del tàndem **NLOC-1**/**NLOC-2**, negant-li a **BLOC** la possibilitat d'intervenir en insercions ***INS2D***/***INS3D*** encara que siguin ortogonals, que prèviament no se n'hagin fet d'obliqües i que **BLOC** no tingui atributs atípics. En resum, no només ens farem enrera d'aquella estratègia, eliminant de **SEGR-ATRIBS** les dues assignacions **(setq WW ())** afegides en el seu tram final, que s'obria amb la condició **(or OO-4 (not NORM))**, sinó que eliminarem la condició mateixa (així no se'ns escaparà el cas **(and (not OO-4) NORM)**), eliminarem del codi tota referència a **WW** i, per suposat, acabarem de fer-ho a **VAL-ATRIBS-2**:

```
(defun VAL-ATRIBS-2 (/ W1 N M V)
  (setq INI () W ())
  (foreach LA (reverse LLAA)
    (if (not EXT) (setq W1 (car LA) LA (cdr LA)))
    (if (car LA)
        (progn
          (if (and ATREQ (not INI))
              (setq INI (not (prompt "\nVerificar valores de atributo"))))
          (VATR T (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq V (list V))
    (if EXT
        (setq WWAA (append WWAA V))
        (if W1 (setq WWAA-1 (append WWAA-1 V)) (setq WWAA-2 (append WWAA-2 V))))))

(defun SEGR-ATRIBS (/ NORM NLOC-1 NLOC-2 ... C-10 C-11 C-72 C-74 OO-3 OZ VZ FW)
  ..........................................................................
  (if (and (or NL2EX BL2EX) (not (XOR NL1EX NL2EX)) (not (XOR BL1EX BL2EX)))
      (progn
        ........................
        (INICOMMAND)
        (if (and (not NL1EX) NORM)
            (MNLOC)
            (if (not (or BL1EX NORM)) (MBLOC))))
      (progn
        (setq  E (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
               BLOK E OO-4 () NE -1 E1 "" E2 "")
```

```
(while E
       ......................
       (setq E (entnext E)))
(VAL-ATRIBS-2)
(INICOMMAND)
(if FW (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))
(command "REGENT"
         "CECOLOR" "PORCAPA"
         "CELTYPE" "PORCAPA"
         "CELWEIGHT" -1)
(WWAA->BLOC)
(if (or NL1EX NL2EX NORM) (MNLOC))
(if (or BL1EX BL2EX (not NORM)) (MBLOC))
(if OO-4
    (progn
      (if (not (tblsearch "BLOCK" "BLOC_NUL"))
          (MAKEBLOC "BLOC_NUL" ()()))
      (setq J 0)
      (foreach O (reverse OO-4)
        (MAKEBLOC (strcat "ATRIBATIP_" (itoa (setq J (1+ J)))
                          "_DE_" BL) T (list O)))
      (setq OO-4 (length OO-4))))
(if (or BL1EX OO-4) (REDEF-BLOC*S))
(setvar "CECOLOR" COL)
(setvar "CELTYPE" TLIN)
(setvar "CELWEIGHT" GLIN))))
```

Així, de passada, homogeneïtzarem el tractament de les insercions ortogonals fetes amb ***INS2D***/***INS3D*** (identificades per l'expressió **(and NORM-XY NORM-Z)**, a **INSERTOK**, i per la variable **NORM** equivalent, a **SEGR-ATRIBS**) i que corresponen als casos senars a la "taula de veritat" del capítol precedent, eliminant el comportament asimètric que hi havíem detectat: també els casos **1**, **3**, **5**, **9**, **11**, **13**, **15**, **17**, **19**, **21**, **23**, **27** i **29** treballaran amb el tàndem **NLOC-1**/**NLOC-2**, tingui **BLOC** o no atributs atípics, com ja passava en els casos **7**, **25** i **31**.

A més de les retallades descrites, prescindirem de l'assignació **(setq BLOC NLOC-1)** feta des de **MNLOC** i **SEGR-ATRIBS** (les línies que us acabem de mostrar ja acusaven aquesta supressió), cosa que després de l'accés a la segona funció, en **INSERTOK**, ens permetrà de substituir **BLOC** per **NLOC-1**. Tanmateix, mantindrem la transferència **(setq BL BLOC)**, localitzada a **INSERTOK** una mica abans, perquè a **INSERT**** ens cal disposar del nom original de **BLOC** per compondre el dels blocs portadors d'atributs atípics, i tenir-lo dipositat a **BL** és més senzill que limitar-lo a **BLOC**, haver de donar un altre nom al primer argument d'aquesta funció i, en conseqüència, haver de dotar **INSERT*** d'un argument per tal que en el seu àmbit la variable **BLOC** pugui representar valors diversos segons que s'hi accedeixi directament des de **INSERTOK** (el bloc original) o des de **INSERT**** (cadascun dels portadors que hem esmentat). En la mateixa línia conservadora, deixarem les dues versions de **C:DESCOMPOK** tal i com estaven: tot i que, un cop excloses de ***INS2D***/***INS3D*** les insercions directes de **BLOC**, podríem simplificar tant la funció **ATRIBS** com el fragment

```
(if ........................
    (setq EX T EY 2)
    (setq EY 0 LE (entget E*)
          EX (and (= (cdr (assoc 0 LE)) "ATTDEF")
                  (= (cdr (assoc 2 LE)) "BLOQUEOK"))
          EX (if EX EX (if A (AVIS)))))
(while (setq E (if (= K 0) (ATRIBS E T) E) E* (ATRIBS E* ()))
   (if (and EX (wcmatch (cdr (assoc 2 LE)) "*BLOQUEOK"))
       (progn (entdel E) (setq EX ()))
       (RESTAURA))
   (setq K (1+ K) E (ssname SA K) E* (entnext E*)))
```

comú a VERSIÓ 21A++ i VERSIÓ 21B++ de **C:DESCOMPOK**, considerant que amb la condició representada amb punts suspensius queden esgotades totes les possibilitats i sobra l'alternativa **(if ... (setq ...) (setq ...))**, volem que ***DESCOMPOK*** sigui aplicable també a les insercions realitzades amb ***INSERT***; altrament, si a l'usuari només li fos permès d'explosionar-les amb ***DESCOMP***, res no evitaria l'aixafament transversal dels atributs fets amb caràcters oblics. Una altra tema és fins a quin punt tenim l'obligació moral de pal·liar les deficiències d'AutoCAD...

Al marge de la problemàtica dels atributs verificables, hi ha una altre punt en què haurem de desdir-nos d'afirmacions sòlides en aparença i de les decisions que se'n derivaven. Recorrent a una metàfora de la qual s'ha abusat, podríem dir (i no és pas l'intent de mantenir el tipus minimitzant la magnitud de la relliscada) que les lleis que regulen el món subatòmic (el concepte de certesa, per exemple) no sempre coincideixen amb les de la realitat macroscòpica. Així, quan en el penúltim capítol (ENCAIX 3D DE BLOCS AMB ATRIBUTS), a punt de donar l'últim toc a la funció **INSPARCIAL**, fèiem el comentari

> "... De forma que eliminarem l'últim argument de **INSPARCIAL:** això no vol dir prescindir de **COMPROBLIC** sinó que, de ser un paràmetre-valor (el receptacle d'aquest cinquè argument) passa a variable local, amb un valor que només dependrà de l'acompliment de $\mathbf{E_x = E_y = E_z}$. Però atenció: no ens deixem endur per la rutina i cometem l'error d'escriure
>
> **(setq COMPROBLIC (not (and (equal EX EY Q0) (equal EX EZ Q0))))**
>
> perquè això no té res a veure amb el grau de precisió que vulguem introduir en el procés, mitjançant la variable **Q0**, sinó que és una condició que detecta AutoCAD amb la seva pròpia precisió de càlcul (remarquem, a més, que no n'hi ha prou amb la igualtat dels tres valors absoluts, sinó que el signe també és significatiu); així, que haurem de fer
>
> **(setq COMPROBLIC (not (= EX EY EZ)))"**

no teníem present un fenómen curiós, que es produeix en el llindar de la precisió de càlcul amb nombres reals, i que ens va posar sobre la pista d'un altre llindar que certament no té res a veure amb el grau de precisió **Q0** introduït en el procés però que es diferencia clarament del primer. Com que l'autor no és un professional de la informàtica, s'haurà de limitar a descriure-us el fenòmen, sense entrar en la seva justificació computacional, i ho farà seguint el fil del mateix raonament que, a partir del descobriment del primer llindar i de la hipòtesi que era l'única causa del fenòmen, va haver de rectificar en arribar a conclusions contradictòries amb la resposta d'AutoCAD, anant a la cerca d'una explicació plausible: un segon llindar, que anomenarem **K0**, per damunt del qual les diferències entre els factors d'escala d'insercions de bloc es traduïssin en l'efecte d'aixafament que coneixem.

El cas és que, després de la redefinició d'un **BLOC** dotat d'atributs típics escrits amb caràcters oblics, seguida del protocol necessari perquè ***INS2D***/***INS3D*** la tingués en consideració (eliminació de **BLOC-2**), en les insercions **(not NORM-Z)** no passava res d'anormal, però en les insercions **NORM-Z** que depenien de genèrics **BLOC*** creats abans de la redefinició de **BLOC** i actualitzats per la funció **REDEF-BLOC*S** just en utilitzar-los per primer cop després del fet, aquests atributs prenien més alçada, com si haguessin sofert l'acció de **REST-OBLIC** quan no tocava perquè **(= EX EY EZ)**. Amb la funció **INSPARCIAL** sota el punt de mira com a principal sospitosa, i sabent que l'única crida des de **INSERT*** en el cas **NORM-Z** (tret de les corresponents als blocs portadors d'atributs atípics, des de **INSERT****) també **(= EX EY EZ)** perquè era **(INSPARCIAL (setq K (/ OX* OX**)) K K X*)**, vam inserir a l'inici d'aquesta funció el testimoni subratllat en el fragment següent:

```
(defun INSPARCIAL (EX EY EZ ANG / COMPROBLIC)
  (command "INSERT" BLOC-1 0 "XYZ" EX EY EZ ANG)
  (if (= BLOC-1 (strcat BLOC " SENSE ATRIBS")) (setq I1 (entget (entlast))))
  (setq SS (ssadd (entlast))
        COMPROBLIC (not (= EX EY EZ)))
  ..........................................
```

Els valors associats als codis 41, 42 i 43 de la llista **I1** no només eren iguals en aparença (limitant-nos als quatre decimals visualitzats) sinó que, si per estar-ne segurs fèiem **(= (cdr (assoc 41 I1)) (= (cdr (assoc 42 I1)) (= (cdr (assoc 43 I1)))** n'obteníem la confirmació. Però vejam què passava a **INSERT*** una mica més endavant, quan avaluant **(command "SCP" "Z" 0 X** "BLOQUE" BLOC* 0 SS "")** definíem **BLOC*** amb la inserció de **BLOC-1** i els atributs resultants de la descomposició de **BLOC-2**: si suposem que **BLOC*** es deia *B_866* i, executada ***INS2D*** o ***INS3D***, escrivíem l'expressió **(setq I2 (entget (cdr (assoc -2 (tblsearch "BLOCK" "B_866")))))**, la nova llista **I2** seguia mostrant uns valors de codi 41, 42 i 43 aparentment iguals; però si fèiem **(= (cdr (assoc 41 I2)) (= (cdr (assoc 42 I2)) (= (cdr (assoc 43 I2)))** per tal de confirmar-ho, l'expressió donava **T** unes vegades i **nil** unes altres (aleatòriament, perquè repetíem el procés amb dades idèntiques). Per saber l'ordre de magnitud de les desigualtats, sols calia escriure **(- (cdr (assoc 41 I2)) (cdr (assoc 42 I2)))** o **(- (cdr (assoc 41 I2)) (cdr (assoc 43 I2)))**: el resultat tan aviat era **0.0** com **±2.22045 e−016**. El perquè de tot plegat ja és matèria en què l'autor (que, a tot

estirar, és un modest programador amateur) no pot pronunciar-se: si ja som en el llindar de la precisió de càlcul amb nombres reals; si, quan un objecte AutoCAD (la inserció de **BLOC-1**, en aquest cas) passa a ser component d'un bloc (de **BLOC***), els seus paràmetres perden precisió; si tot és conseqüència del canvi de sistema de referència (el **(command "SCP" "Z" O X** ...)** previ a la constitució de **BLOC***), ja són coses que hauria d'aclarir un expert. Sí que era al nostre abast, en canvi, un cop detectada l'anomalia, localitzar en **REDEF-BLOC*S** el mecanisme que provocava el creixement en alçada dels atributs. Abans però, justificaríem perquè l'anomalia detectada només tenia transcendència en passar a aquesta funció i no desencadenava cap desgràcia en la mateixa **INSERT*** quan era **(not NORM-Z)** i encara s'accedia una segona vegada a **INSPARCIAL**, després d'haver-hi estat en condicions **(= EX EY EZ)**. Molt senzill: el segon cop (inserció de **BLOC-1*** i **BLOC-2***, explosionant l'última per formar **BLOC****) l'expressió **(setq COMPROBLIC (not (= EX EY EZ)))** era **T** perquè, si més no, **EX** i **EZ** eren diferents per causa del factor compensatori **(/ OX* OXY*)**, que només podia valer **1** en les condicions **NORM-Z** (recordeu que, fora d'aquestes, **(setq XY (list OX* 0 0))**). Pel que fa als atributs atípics escrits amb caràcters oblics, quan prescrivíssim el remei que guariria els típics de la malformació que patien, veuríem perquè els primers n'estaven immunitzats.

Pel que feia a **REDEF-BLOC*S**, una vegada actualitzats tots els **BLOC-1***s i **BLOC-2***s preexistents (**BLOC-1** i **BLOC-2** ja ho havien estat a **SEGR-ATRIBS**) a partir d'un **BLOC** en què s'havia seguit el protocol establert perquè la seva redefinició fos admesa per ***INS2D*/*INS3D***, i arribada l'hora d'actualitzar els **BLOC***s a partir de **BLOC-1** i **BLOC-2**, i els **BLOC****s a partir dels **BLOC-1***s i **BLOC-2***s, estàvem convençuts que la fallada descrita s'explicava per la desviació de **±2.22045 e–016** entre uns factors d'escala que suposàvem d'idèntic valor i que concorrien sobre l'accés a **INSPARCIAL** que, per tal d'evitar-li al lector tornar a la VERSIÓ 19+ del capítol precedent, figura en el fragment que reproduïm:

```
(defun REDEF-BLOC*S (/ LB NZ)
  (setvar "CLAYER" "0")
  ......................
  (if BL1EX
      (progn
        (setq J (1+ (strlen BLOC-1)) NZ NORM-Z NORM-Z ())
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK")) ...)
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat N "*"))
                   (progn
                     (setq E (cdr (assoc -2 LB)) LE (entget E)
                           BLOC-1 (cdr (assoc 2 LE))
                           E (substr BLOC-1 J) NORM-Z (= E "")
                           BLOC-2 (strcat BLOC-2* E))
                     (if (not NORM-Z) (command "SCP" "EZ" O (cdr (assoc 210 LE))))
                     (INSPARCIAL (cdr (assoc 41 LE))
                                 (cdr (assoc 42 LE))
                                 (cdr (assoc 43 LE))
                                 (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (if (not NORM-Z) (command "SCP" "PR"))
                     (command "BLOQUE" BLOC* O SS "")
                     (if OO-4 (command "BLOQUE" (strcat "ATRIBSATIPS_DE_" BLOC*)
                                                O SA ""))))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* NORM-Z NZ))) ...)
```

La identificació era encertada, però no qualificar de condició suficient allò que només era una de les causes (condició necessària). En concret, acceptàvem haver pecat d'ingenuïtat en voler unificar la reconstrucció de tots els **BLOC***s (**NORM-Z**) i **BLOC****s (**(not NORM-Z)**) per simplificar el codi: en tots dos casos recuperàvem els factors d'escala del primer component (la inserció de **BLOC-1** en **BLOC*** i la de **BLOC-1*** en **BLOC****) extrets de les subllistes de codi 41, 42 i 43, quan pel que fa al primer no calia que ens haguéssim molestat, sabent que l'escalat era uniforme i que podíem limitar-nos a un dels tres. Així, les petites diferències aparegudes a **INSERT*** en crear alguns **BLOC***s provocarien que l'actualització fos un esguerro, en arribar a **INSPARCIAL** (ara sí) uns valors no coincidents, activar l'avís **COMPROBLIC** i propiciar una intervenció improcedent de **REST-OBLIC**...

Però pareu esment amb la narració que acabem de fer, perquè si **INSPARCIAL** compleix escrupolosament la seva missió, passarà tot just el contrari: si **COMPROBLIC** s'ha activat era perquè es donàven les condicions per a l'aixafament dels atributs fets amb caràcters oblics, és a dir que **COMPROBLIC** haurà fet el que calia. I, si els atributs manipulats tenen l'alçada majorada en relació als altres components del bloc serà que no s'hauran aixafat. O, si ho voleu més clar, que el dispositiu per a la detecció de l'aixafament s'ha disparat sense que l'efecte s'hagués produït. En definitiva, que **(not (= EX EY EZ))** detecta les desviacions **±2.22045 e—016** però ni aquestes ni els múltiples **±4.44089 e—016** o **±6.66134 e—016** (que poden assolir-se per acumulació en aquesta primera redefinició homologada de **BLOC**, i que entrarien en joc en una segona redefinició) arriben a provocar l'aixafament. Per curar-nos en salut podríem considerar un màxim de **10** redefinicions posteriors a l'existència d'uns genèrics **BLOC***, i substituir a **INSPARCIAL** la condició

```
  (setq ... COMPROBLIC (not (= EX EY EZ)))
```

per una altra que jugués amb la tolerància **10 × 2.22045 e—016**

```
  (setq ... COMPROBLIC (not (and (equal EX EY 2.22045 e—015)
                                 (equal EY EZ 2.22045 e—015)
                                 (equal EZ EX 2.22045 e—015))))
```

Ara bé, per adoptar una solució evasiva potser fóra més net deixar **INSPARCIAL** com estava i renunciar a la simplificació de **REDEF-BLOC*S** esmentada, diversificant-ne l'accés des de cada genèric, segons que sigui del tipus **BLOC*** (compost per una inserció de **BLOC-1** i pels atributs procedents de **BLOC-2**) o del tipus **BLOC**** (per una inserció de **BLOC-1*** i pels atributs procedents de **BLOC-2***):

```
(defun REDEF-BLOC*S (/ LB NZ)
  (setvar "CLAYER" "0")
  ......................
  (if BL1EX
      (progn
        (setq J (1+ (strlen BLOC-1)) NZ NORM-Z NORM-Z ())
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK")) ...)
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat N "*"))
                   (progn
                     (setq E (cdr (assoc -2 LB)) LE (entget E)
                           K (cdr (assoc 41 LE))
                           BLOC-1 (cdr (assoc 2 LE))
                           E (substr BLOC-1 J) NORM-Z (= E "")
                           BLOC-2 (strcat BLOC-2* E))
                     (if (not NORM-Z) (command "SCP" "EZ" O (cdr (assoc 210 LE))))
                     (INSPARCIAL K (if NORM-Z K (cdr (assoc 42 LE)))
                                 (if NORM-Z K (cdr (assoc 43 LE)))
                                 (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (if (not NORM-Z) (command "SCP" "PR"))
                     (command "BLOQUE" BLOC* O SS "")
                     (if OO-4 (command "BLOQUE" (strcat "ATRIBSATIPS_DE_" BLOC*)
                                                O SA ""))))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* NORM-Z NZ)))
  ..........................................................................
```

En realitat, no hauríem fet més que repetir la fórmula usada en el fragment final, que també reproduïrem, i que s'encarrega de l'actualització dels blocs portadors d'atributs atípics (limitar-se a copiar el factor d'escala $\mathbf{E_x}$ associat al codi 41, aplicant-lo també als factors $\mathbf{E_y}$ i $\mathbf{E_z}$). Al cap i a la fi, aquesta innocent llicència era allò que havia preservat els atributs atípics de compartir el destí dels típics en la redefinició:

```
  ..........................................................................
  (if OO-4
      (progn
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2 OO-3 OO-2 OO-2 T NZ OO-4 OO-4 () J 0)
        (repeat NZ
                (setq J (1+ J) BLOC-2 (strcat "ATRIBATIP_" (itoa J) "_DE_" BLOC))
                (tblnext "BLOCK" T)
```

```
                (while (setq LB (tblnext "BLOCK"))
                       (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                                    (strcat BLOC-2 M))
                           (progn
                             (setq BLOC-1 "BLOC_NUL"
                                   E (cdr (assoc -2 LB))
                                   LE (entget E)
                                   K (cdr (assoc 41 LE)))
                             (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                             (command "BLOQUE" BLOC* O SS "")))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* OO-2 OO-3 OO-4 NZ)))
  (setvar "CLAYER" CAPA))
```

Però havent quedat palès que hi ha d'haver un altre llindar d'un ordre de magnitud superior a **±2.22045 e–016**, en contra d'allò que havíem pressuposat en el penúltim capítol, ens veiem en l'obligació de descobrir-lo i tenir-lo present en totes les instàncies on pogués convertir-se en font de conflictes. Ben a prop de **±1.0 e–010**, no ha costat massa de localitzar: amb una desviació de **±9.99994 e–011** entre els factors d'escala, encara no hi ha aixafament; però passant a **±9.99995 e–011**, sí. Així que farem **(setq K0 9.99994e-11 ...)**, just al començar **INSERTOK** i **C:DESCOMPOK**, canviant a **INSPARCIAL** l'assignació

```
  (setq ... COMPROBLIC (not (= EX EY EZ)))
```

per

```
  (setq ... COMPROBLIC (not (and (equal EX EY K0)
                                 (equal EY EZ K0)
                                 (equal EZ EX K0))))
```

i canviant a **COMPR-OBLIC** l'expressió

```
  (not (= (cdr (assoc 41 LE)) (cdr (assoc 42 LE)) (cdr (assoc 43 LE))))
```

per

```
  (setq EX (cdr (assoc 41 LE))
        EY (cdr (assoc 42 LE))
        EZ (cdr (assoc 43 LE)))
  (not (and (equal EX EY K0) (equal EY EZ K0) (equal EZ EX K0)))
```

En un moment donat del penúltim capítol (ENCAIX 3D DE BLOCS AMB ATRIBUTS), crèiem necessari cridar l'atenció del lector per si se li havia passat per alt el fet que no tots els genèrics feien la mateixa mida: les figures de referència de **BLOC*** o del tàndem **BLOC-1***/**BLOC-2*** eren cubs en què l'aresta tenia una longitud $\frac{1}{\cos \alpha_1}$ en el primer cas i $\frac{1}{\cos \beta_1}$ en el segon; les de **BLOC**** eren prismes oblics en què la base quadrada feia $\frac{1}{\cos \alpha_1}$ de costat. No semblava preocupar-nos massa perquè, tot i haver deixat una porta oberta en preguntar-nos si valia la pena de renunciar a la simplicitat conceptual d'uns genèrics unitaris (associats a un cub **1 × 1 × 1** o a un prisma oblic de base **1 × 1**), seguíem convençuts que treballar amb aquests genèrics ens reportava un estalvi de càlcul considerable. En realitat tot venia del capítol NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, on descobríem que, inserint **BLOC** amb $\mathbf{E_{1x} = E_{1y} = E_{1z} = 1}$ per crear un **BLOC*** d'iguals dimensions, la inserció final per assolir l'encaix amb el paral·lelogram **X–O–Y** l'havíem de fer emb els factors d'escala $\mathbf{E_{2x} = O–X'}$, $\mathbf{E_{2y}} = \frac{\mathbf{Y''–Y}}{\mathbf{O–X''}}$. Tanmateix per crear un **BLOC*** que fos $\frac{\mathbf{O–X'}}{\mathbf{O–X''}} \times \frac{\mathbf{O–X'}}{\mathbf{O–X''}}$ vegades **BLOC** calia aplicar $\mathbf{E_{1x} = E_{1y} = E_{1z}} = \frac{\mathbf{O–X'}}{\mathbf{O–X''}}$, però la inserció final era tan simple com $\mathbf{E_{2x} = O–X_1''}$, $\mathbf{E_{2y} = Y_1''–Y_1}$. I el raonament que ens feia decantar cap a la segona opció era irrebatible: cada genèric només serà creat una vegada, però cal suposar que en serà utilitzat força més (si no ja podríem plegar, perquè tota la problemàtica dels genèrics i les complicacions que està arrossegant articular-la de forma coherent no tenia cap altre fonament que aquesta pressumpció); en vista d'això, el que comptava era l'estalvi d'operacions a l'hora de la inserció final. Però cal reconèixer que, en passar a 3D, hem donat per bona la superioritat dels genèrics no unitaris des d'aquesta òptica de facilitat de càlcul, sense molestar-nos a comprovar si de debò era així. Doncs no ho era, com de seguida podreu veure.

De tota manera, el doble reconeixement de culpa (no és només que el procés estigui llastat innecessàriament per operacions aritmètiques, sinó la immerescuda pallissa que ha rebut el sofert lector en el penúltim capítol, a compte d'aquests genèrics infames) no és el pitjor, perque hi ha quelcom de més transcendent: el sistema de genèrics té poca fiabilitat, sobretot quan α_1 i β_1 s'acosten a **90°**; en aquestes condicions, l'aplicació inaugural (la que dóna lloc a l'eclosió del genèric) té un ajust perfecte amb el paral·lelogram d'encaix, perquè la determinació dels factors d'escala es basa en la mètrica d'aquesta figura, pero entre l'arrodoniment dels valors $\frac{1}{\cos \alpha_1} = \frac{\text{o}-\text{x}'}{\text{o}-\text{x}''}$ o $\frac{1}{\cos \beta_1} = \frac{\text{o}-\text{xy}'}{\text{o}-\text{xy}''}$, en passar a formar part de l'identificador del genèric, i la discretització deguda a **Q0**, les insercions que posteriorment el reclamin poden tenir un ajust deficient en relació als propis requisits d'encaix, que en la zona assenyalada poden diferir considerablement de les implícites en el genèric. La diferència és molt més notòria per inadequació de la grandària que per l'angle de gir (de les fallades degudes a l'orientació en parlarem més endavant): recordem només que la funció secant (inversa del cosinus) té una asíntota a **90°**, raó per la qual un petit increment angular pot donar lloc a un fort increment de la secant (que s'hauria de traduir en genèrics sensiblement més grans o petits, segons que l'angle s'acostés o allunyés de l'angle crític); amb **Q0 = 0,001**, copsem com s'enlairarà l'aresta de **BLOC***, considerant que *B_998* encara fa **1**, *B_500* fa **2**, *B_200* fa **5**, *B_050* fa **20**, *B_020* fa **50**, *B_005* fa **200** i *B_002* fa **500** (precisament a partir de *B_020*, que correspon a **88,85°**, les deficiències d'encaix comencen fer-se visibles). Anem doncs al sistema unitari de genèrics, del qual mai no ens hauríem hagut d'apartar.

Tornant a la notació més rigorosa, la primera inserció de **BLOC-1**/**BLOC-2** la farem sense escalat (**$E_x = E_y = E_z = 1$**) i obtindrem **BLOC-1***/**BLOC-2*** que encara mantindran com a referència un cub **1 × 1 × 1** (fins ara l'havíem fet amb els factors **$E_x = E_y = E_z = \frac{\text{o}-(\text{xy}')'}{\text{o}-(\text{xy}')''}$**, i n'obteniem uns genèrics on el cub de referència feia $\frac{\text{o}-(\text{xy}')'}{\text{o}-(\text{xy}')''} \times \frac{\text{o}-(\text{xy}')'}{\text{o}-(\text{xy}')''} \times \frac{\text{o}-(\text{xy}')'}{\text{o}-(\text{xy}')''}$). Definits **BLOC-1***/**BLOC-2*** en el sistema "B2", passarem al "B" per inserir-los transformant el punts $(\text{xy}'_u)'$ i $(\text{z}'_u)'$ en (xy'_u) i (z'_u), i acabarem en el "A2" per definir **BLOC****. Centrant-nos però en la inserció, on ara hem de tenir present que **$\text{o}-(\text{z}'_u)' = \text{o}-(\text{xy}'_u)' = \text{o}-(\text{xy}'_u) = 1$**, detallarem les coordenades

$(\text{xy}'_u)'$ ($\frac{\text{o}-(\text{xy}')''}{\text{o}-(\text{xy}')'}$, $\frac{(\text{xy}')'-(\text{xy}')''}{\text{o}-(\text{xy}')'}$) que cal transformar en (xy'_u) ($\frac{\text{o}-(\text{xy}')''}{\text{o}-(\text{xy})}$, $\frac{(\text{xy}')-(\text{xy}')''}{\text{o}-(\text{xy})}$)

$(\text{z}'_u)'$ ($\frac{\text{o}-(\text{z}')''}{\text{o}-(\text{xy}')'}$, $\frac{(\text{z}')'-(\text{z}')''}{\text{o}-(\text{xy}')'}$) que cal transformar en (z'_u) ($\frac{\text{o}-(\text{z}')''}{\text{o}-(\text{xy}')}$, $\frac{(\text{z}')-(\text{z}')''}{\text{o}-(\text{xy}')}$)

Els factors d'escala seran:

$$\text{E}_x = \frac{\frac{\text{o}-(\text{xy}')''}{\text{o}-(\text{xy}')}}{\frac{\text{o}-(\text{xy}')''}{\text{o}-(\text{xy}')'}} = \frac{\frac{\text{o}-(\text{z}')''}{\text{o}-(\text{xy}')}}{\frac{\text{o}-(\text{z}')''}{\text{o}-(\text{xy}')'}} = \frac{\text{o}-(\text{xy}')'}{\text{o}-(\text{xy}')} = \sin \beta_1 \qquad \text{E}_y = \frac{\frac{(\text{xy}')-(\text{xy}')''}{\text{o}-(\text{xy}')}}{\frac{(\text{xy}')'-(\text{xy}')''}{\text{o}-(\text{xy}')'}} = \frac{\frac{(\text{z}')-(\text{z}')''}{\text{o}-(\text{xy}')}}{\frac{(\text{z}')'-(\text{z}')''}{\text{o}-(\text{xy}')'}} = \frac{\sin \beta_2}{\sin \beta_1}$$

Com que **$\text{o}-(\text{xy}'_u)' = \text{o}-(\text{xy}'_u) = 1$**, cosa que vol dir que en la direcció ***X*** del sistema "B" no hi ha hagut escalat, no caldrà aplicar el factor de compensació que fins ara utilitzàvem: simplement serà **$E_z = 1$**.

Que un **BLOC**** creat així és genèric per a tots els paral·lelepípedes d'encaix en què els sistemes "B2" i "B" estiguin orientats igual respecte a "A2", queda palès en la constància de β_1 i β_2 en aquestes condicions (lògicament, també suposem que l'angle γ_1 és el mateix). La figura de referència de **BLOC**** és un prisma oblic de base quadrada **1 × 1** en què l'altura segueix sent específica del paral·lelepípede d'encaix i no està normalitzada: com passava fins ara (**BLOC-1***/**BLOC-2*** s'inserien

amb els factors d'escala $\mathbf{E_x = \frac{O\text{–}(XY')''}{O\text{–}(X'')}}$, $\mathbf{E_y = \frac{(Z')\text{–}(Z')''}{O\text{–}(X'')}}$ i $\mathbf{E_z = \frac{O\text{–}(XY')''}{O\text{–}(X'')} \times \frac{O\text{–}(X')}{O\text{–}(XY')}}$, i en resultava un prisma oblic de base $\mathbf{\frac{O\text{–}(X')}{O\text{–}(X'')} \times \frac{O\text{–}(X')}{O\text{–}(X'')}}$), en la inserció final no rep cap escalat ***Z*** en relació a ***X***; només en direcció ***Y*** experimenta un escalat diferencial, tret que s'estigui en la situació **NORM-XY**.

Precisament per això, la inserció de **BLOC**** per encaixar el prisma oblic associat en el paral·lelepípede, la podem considerar en 2D, seguint fil per randa el procés descrit en NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, i limitar-nos després a fer $\mathbf{|E_z| = E_x}$. Només cal considerar que $\mathbf{(X'_u)\text{–}O\text{–}(Y'_u)}$ és la base quadrada de **BLOC**** (en comptes del quadrat de **BLOC***), que transformem en el paral·lelogram **(X)–O–(Y)**, i adoptar la notació que venim utilitzant des que l'anàlisi va passar a 3D. Les coordenades dels punts implicats són

$\mathbf{(X'_u)}$ $\left(\mathbf{\frac{O\text{–}(X'')}{O\text{–}(X')}}, \mathbf{\frac{(X')\text{–}(X'')}{O\text{–}(X')}}\right)$ que cal transformar en $\mathbf{(X)}$ $\left(\mathbf{O\text{–}(X'')}, \mathbf{(X)\text{–}(X'')}\right)$

$\mathbf{(Y'_u)}$ $\left(\mathbf{\frac{O\text{–}(Y'')}{O\text{–}(X')}}, \mathbf{\frac{(Y')\text{–}(Y'')}{O\text{–}(X')}}\right)$ que cal transformar en $\mathbf{(Y)}$ $\left(\mathbf{O\text{–}(Y'')}, \mathbf{(Y)\text{–}(Y'')}\right)$

Els factors d'escala seran, doncs, $\mathbf{|E_z| = E_x = O\text{–}(X')}$ i $\mathbf{E_y = \frac{(X)\text{–}(X'')}{(X')\text{–}(X'')} = \frac{(Y)\text{–}(Y'')}{(Y')\text{–}(Y'')}}$ (fins ara teníem $\mathbf{|E_z| = E_x = O\text{–}(X'')}$ i $\mathbf{E_y = (Y)\text{–}(Y'')}$).

Traduïm els canvis a codi i potser apreciarem millors la simplificació que se'n deriva (de tota manera recordeu que, poca o molta, els canvis són indispensables per evitar els nyaps que es produirien a frec dels **90°**). Totes les substitucions són en **INSERT*** i corresponen a l'ordre ***INSERT***, a la vista o dins de **INSPARCIAL:**

- en comptes de
  (INSPARCIAL (setq K (/ OXY* OXY)) K K X*)**
  ha de posar-hi
  (INSPARCIAL 1 1 1 X*)
- en comptes de
  (INSPARCIAL (setq K (/ OX* OX)) K K X*)**
  ha de posar-hi
  (INSPARCIAL 1 1 1 X*)
- en comptes de
```
(INSPARCIAL (setq J (/ 1 OX**) K (* OXY** J))
            (* (distance Z Z**) J) (* K (/ OX* OXY*)) XY**)
```
  ha de posar-hi
```
(INSPARCIAL (setq J (/ OXY* OX*)) (/ (* (distance Z Z**) J) OXY**) 1 XY**)
```
- en comptes de
```
(eval (append '(command "INSERT" BLOC* O "XYZ" OX** (setq J (distance Y Y**))
                        (setq K (* (if (and NORM-Z (not 2D)) (/ OZ OX*) 1)
                                   OX** I)) X**)
              (if ATREQ WWAA-1)))
```
  ha de posar-hi
```
(eval (append '(command "INSERT" BLOC* O "XYZ" OX*
                        (setq J (/ (* (distance Y Y**) OX*) OX**))
                        (setq K (* (if (and NORM-Z (not 2D)) OZ OX*) I)) X**)
              (if ATREQ WWAA-1)))
```

Les tres esmenes introduïdes podíen ser qualificades d'optimitzacions, amb certa dosi de benevolència. Però la que havíem deixat per a les postres és, si ens estem d'eufemismes, la correcció d'un error sense pal·liatius, que potser fins ara havia passat desapercebut perquè ens dedicàvem a qüestions que no eren tan de manual: ens referim a la manipulació improcedent de la tolerància **Q0** en **INSERTOK**, tant en la prevenció contra alineacions com en la detecció dels casos **NORM-XY** i **NORM-Z**. En ambdues situacions havíem caigut en el mateix parany, com apreciareu comparant el fragment que reproduïm (per evitar-vos anar pàgines enrera) amb la versió que la segueix: ni en l'ús del producte vectorial **UV** (de **O–(X)** per **O–(Y)**) ni en l'ús del producte mixt (producte escalar de **UV** per **O–(Z)**), a l'hora de fixar uns valors assimilables a $\sin \mathbf{0°} = \mathbf{0}$ en el primer i a $\cos \mathbf{90°} = \mathbf{0}$ en el segon, teníem en compte

els mòduls d'aquests vectors, i amb això convertíem **Q0** en patró absolut, excessiu per a longituds petites i insuficient per a longituds grans; i la mateixa omissió cometíem en ignorar les longituds **O–(X)**, **O–(Y)**, **O–(Z)**, **(X)–(Y)**, **(X)–(Z)** i **(Y)–(Z)** a l'hora d'introduir la tolerància **Q0** en l'acompliment de Pitàgores. Aquí teniu el fragment espuri, que veníem arrossegant de fa força versions,

```
(while W
  ..........................................
  (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))
        Z (if 2D (mapcar '+ O UV) Z) OZ (if 2D OX OZ)
        W (if (equal UV '(0 0 0) Q0)
              " e Y están alineados."
              (if (and (not 2D)
                       (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                     (mapcar '* UV (mapcar '- Z O))) 0 Q0))
                  ", Y y Z son coplanarios.")))
  (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
(setq I (expt OX 2) J (expt OY 2) K (expt OZ 2)
      NORM-XY (equal (setq X-Y (expt (distance X Y) 2)) (setq X-O-Y (+ I J)) Q0)
      NORM-Z (or 2D
                 (and (equal (expt (distance X Z) 2) (+ I K) Q0)
                      (equal (expt (distance Y Z) 2) (+ J K) Q0)))
      BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
```

i aquí la versió corregida, que no necessàriament vol dir definitiva

```
(while W
  ..........................................
  (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))
        I (distance '(0 0 0) UV)
        Z (if 2D (mapcar '+ O UV) Z) OZ (if 2D OX OZ)
        W (if (<= I (* OX OY Q0))
              " e Y están alineados."
              (if (and (not 2D)
                       (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                     (mapcar '* UV (mapcar '- Z O)))
                              0 (* I OZ Q0)))
                  ", Y y Z son coplanarios.")))
  (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
(setq I (expt OX 2) J (expt OY 2) K (expt OZ 2)
      NORM-XY (equal (setq X-O-Y (sqrt (+ I J)))
                     (setq X-Y (distance X Y)) (* X-Y Q0))
      NORM-Z (or 2D
                 (and (equal (sqrt (+ I K)) (setq W (distance X Z)) (* W Q0))
                      (equal (sqrt (+ J K)) (setq W (distance Y Z)) (* W Q0))))
      BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
```

Si havíem qualificat de postres aquesta darrera esmena, diguem que encara faltaven cafè, copa i puro. Ara sí va de bo que acabem, perquè el motiu que ens ha forçat a un últim incís es mereix un comentari així de poca-solta: els mínims però molestos canvis que la frenètica successió de versions d'AutoCAD introdueix any rere any. En una nota a peu de pàgina del capítol precedent, a propòsit de la descomposició d'insercions de blocs amb atributs editables justificats amb l'opció ***Medio***, fetes en sistemes de referència diferents de l'universal i aplicant un escalat uniforme, comentàvem que en les versions 2000 i anteriors es produïen desplaçaments respecte a la posició prevista, mentre que a partir de la versió 2004 això ja no passava. En **C:DESCOMPOK** (que en aquella ocasió era on podia esdevenir-se el fenòmen i calia actuar en conseqüència), el criteri **(> (atoi (substr (getvar "ACADVER") 1 2)) 15)** ens donava la pista de la versió utilitzada per l'usuari. Ara hem descobert que en **C:INS2D** i **C:INS3D** també es donen respostes diverses segons que la versió d'AutoCAD utilitzada sigui la 15 (2000) o la 16 (2004) i, com que en l'ocasió present caldrà aplicar el test més d'una vegada, en les assignacions inicials de **INSERTOK** optarem per crear una variable **V>15** amb aquest mateix significat.

La nova disparitat de comportaments l'hem localitzada en situacions en què validem per a ***INS2D***/***INS3D*** els canvis introduïts a **BLOC** per redefinicions practicades quan ja s'havien efectuat insercions simples, dobles o triples des d'aquesta aplicació, mitjançant el protocol establert (eliminació de **BLOC-2** o **NLOC-2**, amb ***LIMPIA***), i se

centra en l'abast de la funció estàndard **ENTMAKE** (reclamada des de **MAKEBLOC**, que al seu torn és utilitzada des de **SEGR-ATRIBS** per **MBLOC** i **MNLOC**, per fabricar els tàndems substituts **BLOC-1**/**BLOC-2** i **NLOC-1**/**NLOC-2** respectivament), amb el benentès que ens referim a la redefinició de blocs creats prèviament, no pas al primer cop:
- En AutoCAD 2000 i versions precedents, **ENTMAKE** és autosuficient per visualitzar en les insercions lliures del bloc (és a dir, les que no formen part d'un altre bloc) els efectes de l'actualització, tret de les limitacions pròpies del cas pel que fa als atributs no Constants. Tanmateix, l'actualització no es propaga a les insercions interiors (és a dir, no abasta les que són components d'un altre bloc), ni tan sols forçant la regeneració del dibuix: cal executar **REDEF-BLOC*S**.
- En AutoCAD 2004 (i suposem que en versions successives, si més no les primeres) La funció **ENTMAKE** no és capaç per ella mateixa d'actualitzar els continguts del bloc en les insercions existents (i no només parlem dels atributs editables sinó de la resta de components), sinó que li cal la regeneració posterior del dibuix. Ara bé: amb el concurs d'aquesta regeneració, l'actualització es propaga fins a les insercions situades dintre d'altres blocs.

De tot això en podríem treure una conclusió precipitada: que a partir d'ACAD 2004 ja no ens cal **REDEF-BLOC*S** per actualitzar els genèrics existents. Però només amb blocs sense atributs podrem prescindir d'aquesta funció: no oblidem que **BLOC-2*** no conté cap inserció de **BLOC-2** sinó els atributs (i punts, si s'escau) que resulten d'explosionar-la; que **BLOC**** no conté cap inserció de **BLOC-2*** a més d'una inserció de **BLOC-1***, sinó els atributs que resulten d'explosionar la primera; que, si hi ha atributs atípics, **BLOC-2**** tampoc no conté cap inserció de **BLOC-2*** sinó els punts resultants d'explosionar-la, i els genèrics de cada bloc portador tampoc no estan formats per la inserció sinó per l'atribut atípic alliberat en l'explosió. En tots aquests casos necessitem **REDEF-BLOC*S** per poder reproduir el procés de generació. Així doncs, de primer caldrà assegurar l'actualització dels únics blocs substituts que es visualitzen, és a dir, de **NLOC-1** i **NLOC-2**, canviant en **SEGR-ATRIBS** la línia

```
    (if (or NL1EX NL2EX NORM) (MNLOC))
```

per

```
(if (or NL1EX NL2EX NORM) (progn (MNLOC) (if V>15 (command "REGENT"))))
```

No cal que fem el mateix amb els blocs portadors d'atributs atípics (que en casos determinats s'insereixen directament), perquè no en treuríem res: recordeu que les redefinicions d'un bloc no afecten els atributs N o P de les insercions existents, ni tan sols pel que fa a la posició.

Més endavant, si volem prescindir de **REDEF-BLOC*S** quan sigui **V>15** i **BLOC** no tingui atributs editables, substituint-la per una regeneració, podríem canviar la línia

```
    (if (or BL1EX OO-4) (REDEF-BLOC*S))
```

per

```
(if (and (or BL1EX OO-4) (not (and V>15 (not OO-2) (not OO-4))))
    (REDEF-BLOC*S)
    (command "REGENT"))
```

tot i que simplificarem la condició fent

```
    (if (and (or BL1EX OO-4) (or (not V>15) OO-2 OO-4))
        (REDEF-BLOC*S)
        (command "REGENT"))
```

Però si el dibuix consta de molts objectes convé no prodigar-se en regeneracions, raó per la qual prescindirem de la primera substitució i la segona la deixarem en

```
    (if (and (or BL1EX OO-4) (or (not V>15) OO-2 OO-4))
        (progn
          (if (and V>15 (or NL1EX NL2EX NORM)) (command "REGENT"))
          (REDEF-BLOC*S))
        (command "REGENT"))
```

perquè cal tenir en compte que **(and V>15 (or NL1EX NL2EX NORM))** no és incompatible amb **(not (and (or BL1EX OO-4) (or (not V>15) OO-2 OO-4)))** i que quan s'acomplissin ambdues condicions hi hauria dues regeneracions del dibuix.

Com que des de la VERSIÓ 19+ hem estat abusant de presentacions fragmentàries del codi, aquest cop serà convenient d'oferir el contingut complet de la VERSIÓ 22++, circumstància que aprofitarem per posar una mica d'ordre en el paper inicialment atribuït a les variables **M** i **N** de **INSERTOK**, que en el decurs de les successives versions havia anat quedant encotillat i convenia readaptar:

```
; VERSIÓ 22++

;;; (vl-load-com)
;;; (vlr-remove-all)
;;; (vlr-COMMAND-reactor ()'((:vlr-CommandWillStart . -ECO-1)
;;;                          (:vlr-CommandEnded . -ECO-2)))
```

```
;;; (defun -ECO-1 (N-REACTIU L-ORDRE)
;;;   (if CTRL--ECO (BS (1+ (strlen (getcname (strcat "_" (car L-ORDRE))))))))

;;; (defun -ECO-2 (N-REACTIU L-ORDRE) (if CTRL--ECO (BS (1+ (strlen CTRL--ECO)))))

;;; (defun SENSE-RASTRE ()
;;;   (command "DESHACER" (progn (if (= ECO 1)
;;;                                  (progn (BS 100) (setq CTRL--ECO "I")))
;;;                              "I"))
;;;   (if (= ECO 1) (progn (BLANC) (setq CTRL--ECO ()))))

(defun QUASI-SENSE-RASTRE ()
  (command "DESHACER" (progn (if (= ECO 1) (BS 100)) "I"))
  (if (= ECO 1) (BLANC)))

(defun C:INS2D () (INSERTOK T))

(defun C:INS3D () (INSERTOK ()))

(defun BS (I) (repeat I (princ "\10 \10")))

(defun BLANC () (princ "\r") (repeat 100 (princ " ")) (princ "\r") (princ))

(defun REFX ()
  (strcat "\"" BLOC "\"\nNo se puede insertar con INSERT ni INSERTOK\nporque "
          (if (= (logand (cdr (assoc 70 O)) 16) 0) "es" "depende de")
          " una referencia externa."))

(defun RUTES (/ MS PREFIX C)
  (setq MS (strcat "\"" BLOC
             ".dwg\"\nNo se encuentra el archivo en el camino de búsqueda:\n  "
             (substr (findfile "ACAD.EXE") 1 (- (strlen (findfile "ACAD.EXE")) 8))
             " (directorio actual)\n  " (getvar "DWGPREFIX") "\n  ")
        PREFIX (getvar "ACADPREFIX") N 0)
  (repeat (strlen PREFIX)
          (setq N (1+ N) C (substr PREFIX N 1)
                MS (strcat MS (if (= C ";") "\n  " C))))
  (substr MS 1 (- (strlen MS) 3)))

(defun DIBUIX (/ A B)
  (command "INSERT" BLOC ())
  (setq A (tblnext "BLOCK" T))
  (while (setq B (tblnext "BLOCK")) (setq A B))
  (strcase (cdr (assoc 2 A))))

(defun DEFECTE (TXT) (if (= TXT "") "" (strcat " <" TXT ">")))

(defun U*V (U V)
  (list (- (* (cadr U) (last V)) (* (last U) (cadr V)))
        (- (* (last U) (car  V)) (* (car  U) (last V)))
        (- (* (car  U) (cadr V)) (* (cadr U) (car  V)))))

(defun VATR (W* N* M* VD*)
  (setq W W* N N* M M* VD VD*
        V (if ATREQ
              (getstring (strcat "\n" (if (= M "")
                                          (if (wcmatch N "VRF`**") (substr N 5) N)
                                          M)
                                 (DEFECTE VD) ": ") T)
              "")
        V (if (= V "") VD V)))

(defun VAL-ATRIBS-1 (LA / M V)
  (if (= (logand ICVP 8) 0)
      (progn
        (if (and ATREQ (not INI))
            (setq INI (not (prompt "\nIndique valores de atributo"))))
```

```
        (VATR (wcmatch (cdr (assoc 2 LA)) "VRF`**") (cdr (assoc 2 LA))
              (cdr (assoc 3 LA)) (cdr (assoc 1 LA)))
        (setq LLAA (cons (list W N M V) LLAA)))))

(defun VAL-ATRIBS-2 (/ W1 M V)
  (setq INI () W ())
  (foreach LA (reverse LLAA)
    (if (not EXT) (setq W1 (car LA) LA (cdr LA)))
    (if (car LA)
        (progn
          (if (and ATREQ (not INI))
              (setq INI (not (prompt "\nVerificar valores de atributo"))))
          (VATR T (cadr LA) (caddr LA) (last LA)))
        (setq V (last LA)))
    (setq V (list V))
    (if EXT
        (setq WWAA (append WWAA V))
        (if W1 (setq WWAA-1 (append WWAA-1 V)) (setq WWAA-2 (append WWAA-2 V))))))

(defun VAL-ATRIBS (OO EXT E1E2)
  (foreach O (reverse OO)
    (setq ICVP (cdr (assoc 70 O)))
    (if (and E1E2 (or (not O) (= (logand ICVP 8) 8)))
        (setq LLAA (cons () LLAA))
        (VAL-ATRIBS-1 O)))
  (VAL-ATRIBS-2))

(defun MAKEBLOC (BLOC/2 ATRIBS OO)
  (entmake (list '(0 . "BLOCK") (cons 2 BLOC/2) '(10 0 0 0)
                 (cons 70 (if ATRIBS 2 0))))
  (foreach O (reverse OO) (entmake O))
  (entmake '((0 . "ENDBLK"))))

(defun MNLOC ()
  (MAKEBLOC NLOC-1 (or AT-C OO-2) (append OO-2 OO-1))
  (MAKEBLOC NLOC-2 () OO-3))

(defun MBLOC ()
  (MAKEBLOC BLOC-1 AT-C OO-1)
  (MAKEBLOC BLOC-2 OO-2 (append OO-3 OO-2)))

(defun LINIA-BASE ()
  (polar (cdr (assoc 11 LE))
         (- (cdr (assoc 50 LE)) PI/2)
         (* (cdr (assoc 40 LE))
            (if (= C-74 1)
                (/ 1.0 -3)
                (if (= C-74 2) 0.5 1)))))

(defun PUNT (P) (list '(0 . "POINT") (cons 10 (trans P VZ 0)) (cons 210 VZ)))

(defun REST-OBLIC ()
  (setq A (cos (cdr (assoc 51 (entget E*)))))
  (if (not (equal A 1 Q0))
      (progn
        (setq LE (entget E)
              LE (subst (cons 40 (/ (cdr (assoc 40 LE)) A))
                        (assoc 40 LE) LE)
              LE (subst (cons 41 (* (cdr (assoc 41 LE)) A))
                        (assoc 41 LE) LE))
        (entmod LE))))

(defun INSPARCIAL (EX EY EZ ANG / COMPROBLIC)
  (command "INSERT" BLOC-1 O "XYZ" EX EY EZ ANG)
  (setq SS (ssadd (entlast))
        COMPROBLIC (not (and (equal EX EY K0)
                             (equal EY EZ K0)
                             (equal EZ EX K0))))
```

```
  (if (or OO-2 OO-4)
      (progn
        (command "ATTREQ" 0
                 "INSERT" BLOC-2 O "XYZ" EX EY EZ ANG
                 "ATTREQ" (if ATREQ 1 0)
                 "DESCOMP" (entlast))
        (setq SA (ssget "P") K -1)
        (while (setq E (ssname SA (setq K (1+ K))))
               (ssadd E SS))))
  (if (and OO-4 (or NORM-Z COMPROBLIC))
      (progn
        (setq SA (ssadd) K (if NORM-Z (- (sslength SS) (* 3 OO-4)) 1))
        (repeat (* 3 OO-4)
                (setq E (ssname SS K))
                (ssadd E SA)
                (ssdel E SS))))
  (if COMPROBLIC
      (progn
        (setq K 0 E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC-2))))
        (while (setq K (1+ K) E (ssname SS K))
               (REST-OBLIC)
               (setq E* (entnext E*))))))

(defun XOR (A B) (and (not (and A B)) (or A B)))

(defun B->N (A B / C)
  (setq C (= B (if A BLOC-1 NLOC-1))
        E (cdr (assoc -2 (tblsearch "BLOCK" B))))
  (if (/= (cdr (assoc 0 (entget E))) "ENDBLK")
      (while E
             (setq LE (entget E) AT (= (cdr (assoc 0 LE)) "ATTDEF"))
             (if (and AT (or (and A (not C))
                             (and (not A) C
                                  (= (logand (cdr (assoc 70 LE)) 2) 0))))
                 (setq OO-2 (cons LE OO-2))
                 (if C
                     (setq AT-C (if AT-C T AT)
                           OO-1 (cons LE OO-1))
                     (setq OO-3 (cons LE OO-3))))
             (setq E (entnext E)))))

(defun ATRIB-ATIPIC (I)
  (tblsearch "BLOCK"
             (strcat "ATRIBATIP_" (itoa (set I (1+ (eval I)))) "_DE_" BLOC)))

(defun BLOC->WWAA (/ EE OO EEOO EO OORD)
  (setq OO (reverse OO-1)
        E1 (car OO) E2 (cadr OO)
        E1 (read (strcat "(" (cdr (assoc 1 E1)) ")"))
        E2 (read (strcat "(" (cdr (assoc 1 E2)) ")")) N -1
        EE (repeat K (setq EE (cons (nth (setq N (1+ N)) E2) EE)))
        E2 (reverse EE)
        EE (append E1 E2) OO () N 0
        OO (repeat K
             (setq OO (cons (entget (cdr (assoc -2 (ATRIB-ATIPIC 'N)))) OO)))
        OO (reverse (append OO OO-2))
        EEOO (mapcar 'cons EE OO) N -1
  (repeat (+ (length OO-2) J)
          (setq N (1+ N) EO (cdr (assoc N EEOO))
                OORD (cons EO OORD)))
  (VAL-ATRIBS OORD T T)
  (foreach E E1
           (if (setq N (nth E WWAA))
               (setq WWAA-1 (append WWAA-1 (list N)))))
  (foreach E E2
           (if (setq N (nth E WWAA))
               (setq WWAA-2 (append WWAA-2 (list N))))))
```

```
(defun WWAA->BLOC (/ O1 O2)
  (setq O1 (last OO-1))
  (if (and (= (cdr (assoc 0 O1)) "ATTDEF")
           (= (cdr (assoc 2 O1)) "WWAA-1")
           (= (cdr (assoc 70 O1)) 3))
      (setq OO-1 (reverse (cdr (cdr (reverse OO-1))))))
  (if OO-4
      (setq O1 (list '(0 . "ATTDEF") '(8 . "0") '(70 . 3) '(2 . "WWAA-1")
                     '(3 . "") (cons 1 E1) '(72 . 0) '(74 . 0) '(10 0 0 0)
                     '(210 0 0 1) '(7 . "Standard") '(40 . 1) '(41 . 1))
            O2 (subst (cons 1 E2) (cons 1 E1)
                      (subst '(2 . "WWAA-2") '(2 . "WWAA-1")
                             O1))
            OO-1 (append OO-1 (list O2 O1)))))

 (defun INICOMMAND ()
  (setq Z (trans Z 1 0)
        Z-XY Y
        Y (trans Y 1 0))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
  (setvar "CMDECHO" 0)
  (command "OSMODE" 0
           "ATTDIA" 0
           "EXPERT" 2
           "SCP" "3" O X Z-XY)
  (setq O '(0 0 0) X (list OX 0 0) Y (trans Y 0 1) Z (trans Z 0 1)
        Z-XY (list (car Z) (cadr Z) 0) I (if (< (last Z) 0) -1 1)))

(defun PRO-DESHACER-I/F ()
  (command "BLOQUE" BLOC-1* O (setq E (ssname SS 0)) "")
  (if (or OO-2 OO-4)
      (command "BLOQUE" BLOC-2* O (ssdel E SS) "")
      (MAKEBLOC BLOC-2* () ())))

(defun REDEF-BLOC*S (/ LB)
  (setvar "CLAYER" "0")
  (setvar "CECOLOR" "PORBLOQUE")
  (setvar "CELTYPE" "PORBLOQUE")
  (setvar "CELWEIGHT" -2)
  (if BL1EX
      (progn
        (setq J (1+ (strlen BLOC-1))
              N NORM-Z NORM-Z ())
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC-1* (cdr (assoc 2 LB)))
                            (strcat BLOC-1 "_" M M))
                   (progn
                     (setq E (cdr (assoc -2 LB))
                           LE (entget E)
                           K (cdr (assoc 41 LE))
                           BLOC-2* (strcat BLOC-2 (substr BLOC-1* J)))
                     (command "SCP" "EZ" O (cdr (assoc 210 LE)))
                     (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                     (command "SCP" "PR")
                     (PRO-DESHACER-I/F))))
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2)
        (tblnext "BLOCK" T)
        (while (setq LB (tblnext "BLOCK"))
               (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                            (strcat BLOC "_" M "*"))
                   (progn
                     (setq E (cdr (assoc -2 LB)) LE (entget E)
                           K (cdr (assoc 41 LE))
                           BLOC-1 (cdr (assoc 2 LE))
                           E (substr BLOC-1 J) NORM-Z (= E "")
                           BLOC-2 (strcat BLOC-2* E))
```

```
                    (if (not NORM-Z) (command "SCP" "EZ" O (cdr (assoc 210 LE))))
                    (INSPARCIAL K (if NORM-Z K (cdr (assoc 42 LE)))
                                  (if NORM-Z K (cdr (assoc 43 LE)))
                                  (/ (* (cdr (assoc 50 LE)) 180) PI))
(if (not NORM-Z) (command "SCP" "PR"))
                    (command "BLOQUE" BLOC* O SS "")
                    (if OO-4 (command "BLOQUE" (strcat "ATRIBSATIPS_DE_" BLOC*)
                                                O SA "")))))
      (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* NORM-Z N)))
  (if OO-4
      (progn
        (setq BLOC-1* BLOC-1 BLOC-2* BLOC-2 OO-3 OO-2 OO-2 T N OO-4 OO-4 () J 0)
        (repeat N
                (setq J (1+ J) BLOC-2 (strcat "ATRIBATIP_" (itoa J) "_DE_" BLOC))
                (tblnext "BLOCK" T)
                (while (setq LB (tblnext "BLOCK"))
                       (if (wcmatch (setq BLOC* (cdr (assoc 2 LB)))
                                    (strcat BLOC-2 "_" M))
                           (progn
                             (setq BLOC-1 "BLOC_NUL"
                                   E (cdr (assoc -2 LB))
                                   LE (entget E)
                                   K (cdr (assoc 41 LE)))
                             (INSPARCIAL K K K (/ (* (cdr (assoc 50 LE)) 180) PI))
                             (command "BLOQUE" BLOC* O SS "")))))
        (setq BLOC-1 BLOC-1* BLOC-2 BLOC-2* OO-2 OO-3 OO-4 N)))
  (setvar "CLAYER" CAPA))

(defun J-PUNTS (LB)
  (setq E (cdr (assoc -2 LB)) J 0)
  (while E (setq J (if (= (cdr (assoc 0 (entget E))) "POINT") (1+ J) J)
                 E (entnext E)))
  J)

(defun AVIS ()
  (alert (strcat "ATENCIÓN:\n\nComo \"" BLOC "\" se creó con la orden BLOQUE "
                 "y no con BLOQUEOK,\n\nlos atributos justificados por su punto "
                 "Medio pueden aparecer movidos")))

(defun AJUSTA-E (E)
  (if (= E "")
      ""
      (progn
        (setq OZ (mapcar '1+ (read (strcat "(" E ")"))) E "")
        (foreach Z OZ (setq E (strcat E " " (itoa Z)))))))

(defun SEGR-ATRIBS (/ NORM NLOC-2 NL1EX NL2EX BL1EX BL2EX BLOK AT AT-C NE E1 E2
                      EXT INI LLAA C-10 C-11 C-72 C-74 OO-3 OZ VZ FW)
  (setq NORM (and NORM-XY NORM-Z)
        NLOC-1 (strcat BLOC "_AMB_ATRIBSTIPS")
        NL1EX (tblsearch "BLOCK" NLOC-1)
        NLOC-2 (strcat "ATRIBSATIPS_DE_" BLOC)
        NL2EX (tblsearch "BLOCK" NLOC-2)
        BLOC-1 (strcat BLOC "_SENSE_ATRIBS")
        BL1EX (tblsearch "BLOCK" BLOC-1)
        BLOC-2 (strcat "ATRIBS_DE_" BLOC)
        BL2EX (tblsearch "BLOCK" BLOC-2) K 0
        OO-4 (while (ATRIB-ATIPIC 'K) K)
        K (if NL2EX
              (J-PUNTS NL2EX)
              (if BL2EX (J-PUNTS BL2EX)))
        K (if (> K 0) (/ K 3)) J OO-4)
  (if (< K OO-4)
      (setq OO-4 K)
      (if (> K OO-4) (setq NL2EX () BL2EX () FW T)))
  (if (and (or NL2EX BL2EX)
           (not (XOR NL1EX NL2EX))
           (not (XOR BL1EX BL2EX)))
```

```
(progn
  (if BL1EX
      (progn
        (B->N T BLOC-1)
        (B->N T BLOC-2))
      (progn
        (B->N () NLOC-1)
        (B->N () NLOC-2)))
  (if (and (setq LE (last OO-2))
           (= (cdr (assoc 2 LE)) "NO-BLOQUEOK"))
      (AVIS))
  (if OO-3
      (BLOC->WWAA)
      (if OO-2
          (progn
            (VAL-ATRIBS OO-2 T ())
            (setq WWAA-1 WWAA))))
  (INICOMMAND)
  (if (and (not NL1EX) NORM)
      (MNLOC)
      (if (not (or BL1EX NORM)) (MBLOC))))
(progn
  (setq  E (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
         BLOK E OO-4 ()
         NE -1 E1 "" E2 "")
  (while E
         (setq LE (entget E))
         (if (equal E BLOK)
             (setq BLOK (if (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                 (= (cdr (assoc 2 LE)) "BLOQUEOK")) 2 0)))
         (if (> BLOK 1)
             (setq BLOK 1)
             (if (and (setq AT (= (cdr (assoc 0 LE)) "ATTDEF"))
                      (= (logand (setq ICVP (cdr (assoc 70 LE))) 2) 0))
                 (progn
                   (setq NE (1+ NE)
                         C-72 (cdr (assoc 72 LE))
                         C-74 (cdr (assoc 74 LE))
                         BLOK (if (and (= BLOK 0) (= C-72 4))
                                  (progn
                                    (setq E1 (AJUSTA-E E1)
                                          E2 (AJUSTA-E E2)
                                          E1 (strcat " 0" E1)
                                          NE (1+ NE)
                                          OO-2
                                          (append OO-2
                                            (list
                                              (list '(0 . "ATTDEF")
                                                    '(8 . "0") '(70 . 9)
                                                    '(2 . "NO-BLOQUEOK")
                                                    '(3 . "") '(1 . "")
                                                    '(72 . 0) '(74 . 0)
                                                    '(10 0 0 0)
                                                    '(210 0 0 1)
                                                    (assoc 7 LE)
                                                    (assoc 40 LE)
                                                    (assoc 41 LE)))))
                                    (AVIS))
                                  BLOK)
                         LE (if (= (logand ICVP 4) 4)
                                (subst (cons 2 (strcat "VRF*"
                                                       (cdr (assoc 2 LE))))
                                       (assoc 2 LE)
                                       (subst (cons 70 (setq ICVP
                                                             (- ICVP 4)))
                                              (assoc 70 LE)
                                              LE))
                                LE)
```

```
                                LE (if (= C-72 4)
                                       (subst (cons 72 (setq C-72 1))
                                              (cons 72 4)
                                              (subst (cons 74 (setq C-74 2))
                                                     (assoc 74 LE)
                                                     LE))
                                       LE)
                                LE (if (and (< C-72 3) (> C-74 0))
                                       (subst (cons 74 0)
                                              (assoc 74 LE)
;;                                            (subst (cons 11 (LINIA-BASE))
;;                                                   (assoc 11 LE)
;;                                                   LE))
;;   Podeu usar les 3 línies precedents com alternativa a les 5 línies subsegüents
                                              (if (> C-72 0)                    ;;
                                                  (subst (cons 11 (LINIA-BASE)) ;;
                                                         (assoc 11 LE)          ;;
                                                         LE)                    ;;
                                                     LE))                       ;;
                                       LE)
                                OZ (last (assoc 10 LE))
                                VZ (cdr (assoc 210 LE)))
                          (VAL-ATRIBS-1 LE)
                          (if (and (equal OZ 0 Q0) (equal VZ '(0 0 1) Q0))
                              (setq LLAA (if (= (logand ICVP 8) 0)
                                             (cons (cons T (car LLAA)) (cdr LLAA))
                                             LLAA)
                                    E1 (strcat E1 " " (itoa NE))
                                    OO-2 (cons LE OO-2))
                              (setq LLAA (if (= (logand ICVP 8) 0)
                                             (cons (cons () (car LLAA)) (cdr LLAA))
                                             LLAA)
                                    E2 (strcat E2 " " (itoa NE))
                                    OO-3 (cons (PUNT (list 0 0 OZ)) OO-3)
                                    OO-3 (cons (PUNT (list 1 0 OZ)) OO-3)
                                    OO-3 (cons (PUNT (list 0 1 OZ)) OO-3)
                                    C-10 (assoc 10 LE)
                                    C-11 (assoc 11 LE)
                                    OO-4 (cons (subst (list 10 (cadr C-10)
                                                               (caddr C-10) 0)
                                                      C-10
                                                      (subst (list 11 (cadr C-11)
                                                                      (caddr C-11) 0)
                                                             C-11
                                                             (subst '(210 0 0 1)
                                                                    (assoc 210 LE)
                                                                    LE)))
                                               OO-4))))
                    (setq OO-1 (cons LE OO-1) AT-C (if AT-C T AT))))
            (setq E (entnext E)))
      (VAL-ATRIBS-2)
      (INICOMMAND)
      (if FW (command "LIMPIA" "B" (strcat NLOC-2 "," BLOC-2) "N"))
      (command "REGENT"
               "CECOLOR" "PORCAPA"
               "CELTYPE" "PORCAPA"
               "CELWEIGHT" -1)
      (WWAA->BLOC)
      (if (or NL1EX NL2EX NORM) (MNLOC))
      (if (or BL1EX BL2EX (not NORM)) (MBLOC))
      (if OO-4
          (progn
            (if (not (tblsearch "BLOCK" "BLOC_NUL")) (MAKEBLOC "BLOC_NUL" ()()))
            (setq J 0)
            (foreach O (reverse OO-4)
                     (MAKEBLOC (strcat "ATRIBATIP_" (itoa (setq J (1+ J)))
                                       "_DE_" BL) T (list O)))
            (setq OO-4 (length OO-4))))
```

```
        (if (and (or BL1EX OO-4) (or (not V>15) OO-2 OO-4))
            (progn
              (if (and V>15 (or NL1EX NL2EX NORM)) (command "REGENT"))
              (REDEF-BLOC*S))
            (command "REGENT"))
        (setvar "CECOLOR" COL)
        (setvar "CELTYPE" TLIN)
        (setvar "CELWEIGHT" GLIN))))

(defun CALCULA (U V OV AGUT U* U** OU* OU** V**)
  (setq A (polar O (- (angle O V) PI/2) OV)
        W (if AGUT (angle A U) (angle U A))
        B (mapcar '/ (mapcar '+ A U) '(2 2 2))
        W (angle O (polar B W (distance O B))))
  (set U* (inters U (polar U W 1) O B ()))
  (set U** (inters O (polar O (+ W PI/2) 1) U (eval U*) ()))
  (set OU* (distance O (eval U*)))
  (set OU** (distance O (eval U**)))
  (set V** (inters V (polar V W 1) O (eval U**) ())))

 (defun FIX+ (O)
  (setq N (itoa (if (> (- O (setq N (fix O))) 0.5) (1+ N) N)))
  (repeat (- (strlen M) (strlen N)) (setq N (strcat "0" N)))
  N)

(defun INSERT** (BLOC X Y Z G / JJ KK SSAA)
  (command "INSERT" (strcat "ATRIBSATIPS_DE_" BLOC) O "XYZ" X Y Z G
           "DESCOMP" (entlast)
           "SCP" "")
  (setq SSAA (ssget "P") JJ 0 KK -1)
  (repeat OO-4
          (setq JJ (1+ JJ) BLOC (strcat "ATRIBATIP_" (itoa JJ) "_DE_" BL)
                LE (entget (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
                ICVP (cdr (assoc 70 LE))
                WWAA-1 (if (= (logand ICVP 8) 0) (list (car WWAA-2)))
                WWAA-2 (if WWAA-1 (cdr WWAA-2) WWAA-2)
                KK (1+ KK) O (cdr (assoc 10 (entget (ssname SSAA KK))))
                KK (1+ KK) X (cdr (assoc 10 (entget (ssname SSAA KK))))
                KK (1+ KK) Y (cdr (assoc 10 (entget (ssname SSAA KK)))))
          (command "SCP" "3" O X Y)
          (setq OX (distance O X)
                OY (distance O Y) OZ OX
                O '(0 0 0) X (list OX 0 0)
                Y (trans Y 0 1) Z (U*V X Y)
                Z-XY O I (if (< (last Z) 0) -1 1)
                BLOC-1 "BLOC_NUL" BLOC-2 BLOC
                NORM-XY (equal (setq X-Y (expt (distance X Y) 2))
                               (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
                NORM-Z T OO-2 T OO-4 ())
          (if NORM-XY
              (eval (append '(command "INSERT" BLOC O OX OY 0) (if ATREQ WWAA-1)))
              (INSERT*))
          (command "SCP" "PR"))
  (command "SCP" "PR"
           "BORRA" SSAA ""))

(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** Z-XY* XY XY* XY** OXY* OXY** ABLOC*)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X
            Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'OX* 'OX** 'Y**)
```

```
      (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
            A (inters Z-XY (polar Z-XY W 1) O X** ())
            Z-XY* (polar A (angle A Z-XY)
                          (/ (* (distance A Z-XY) (distance X* X**))
                             (distance X X**))))))
  (if NORM-Z
      (setq BLOC* (strcat BLOC "_"))
      (progn
        (setq X (inters X* Y* O Z-XY* ()) X (if X X Z-XY*)
              Z-XY (- (angle O X) (angle Y* X*))
              Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
              Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
              Z (trans (list (car Z-XY*) (cadr Z-XY*) (last Z)) 1 0))
        (command "SCP" "Z" O X
                 "SCP" "X" (* I 90))
        (setq XY (list OX* 0 0) Z (trans Z 0 1))
        (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'OXY* 'OXY** 'Z**)
        (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                            (FIX+ (/ OXY** OXY* Q0)))
              BLOC* (strcat BLOC K)
              BLOC-1* (strcat BLOC-1 K) BLOC-2* (strcat BLOC-2 K)
              BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
        (command "SCP" "PR" "SCP" "PR")
        (if (not BL*EX)
            (progn
              (INSPARCIAL 1 1 1 X*)
              (command "SCP" "Z" O X
                       "SCP" "X" 90
                       "GIRA" SS "" O XY*
                       "SCP" "Z" O XY**)
              (PRO-DESHACER-I/F)
              (command "SCP" "PR")))
        (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*
              BLOC* (strcat BLOC* "_" (FIX+ (/ OXY** OX* Q0))))))
  (if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
  (if OO-4 (setq ABLOC* (strcat "ATRIBSATIPS_DE_" BLOC*)))
  (if (or NORM-Z BL*EX) (setq BL*EX (and (tblsearch "BLOCK" BLOC*)
                                         (if OO-4 (tblsearch "BLOCK" ABLOC*) T))))
  (if (not BL*EX)
      (progn
        (if NORM-Z
            (INSPARCIAL 1 1 1 X*)
            (progn
              (if B (command "SCP" "Z" O X "SCP" "X" 90))
              (INSPARCIAL (setq J (/ OXY* OX*))
                          (/ (* (distance Z Z**) J) OXY**) 1 XY**)
              (command "SCP" "PR" "SCP" "PR")))
        (command "SCP" "Z" O X**
                 "BLOQUE" BLOC* O SS "")
        (if OO-4 (command "BLOQUE" ABLOC* O SA ""))
        (command "SCP" "PR")))
  (command "CELWEIGHT" GLIN
           "CELTYPE" TLIN
           "CECOLOR" COL
           "CLAYER" CAPA)
  (eval (append '(command "INSERT" BLOC* O "XYZ" OX*
                          (setq J (/ (* (distance Y Y**) OX*) OX**))
                          (setq K (* (if (and NORM-Z (not 2D)) OZ OX*) I)) X**)
                (if ATREQ WWAA-1)))
  (if OO-4 (INSERT** BLOC* OX** J K X**)))

(defun INSERTOK (2D / K0 Q0 2*PI PI/2 V>15 CAPA COL TLIN GLIN ECO CTRL--ECO OSN
                      ATDIA ATREQ EXPERT UV BL BLOC BLOC* BLOC-1 BLOC-2 BLOC-1*
                      BLOC-2* NLOC-1 BL*EX ICVP WWAA WWAA-1 WWAA-2 WW SA SS O X Y
                      Z Z-XY OX OY OZ X-Y X-O-Y A B E LE E* I J K M N W NORM-XY
                      NORM-Z OO-1 OO-2 OO-4)
  (setq K0 9.99994e-11 Q0 0.001 M "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq M (strcat M "#")))
```

```
  (setq 2*PI (* PI 2) PI/2 (/ PI 2)
        V>15 (> (atoi (substr (getvar "ACADVER") 1 2)) 15)
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        EXPERT (getvar "EXPERT")
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit)) W T)
  (while W
    (initget 1) (setq O (getpoint "\nPrecise punto de inserción: "))
    (foreach P (if 2D '("X" "Y") '("X" "Y" "Z"))
      (initget 1)
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                  " desde el punto de inserción: "))
            W (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC  " <1>: "))
            W (if W W 1))
      (set (read P) (mapcar '(lambda (CO CA) (+ CO (/ (- CA CO) W))) O A))
      (set (read (strcat "O" P)) (/ (distance O A) W)))
    (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))
          I (distance '(0 0 0) UV)
          Z (if 2D (mapcar '+ O UV) Z) OZ (if 2D OX OZ)
          W (if (<= I (* OX OY Q0))
                " e Y están alineados."
                (if (and (not 2D) (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                                (mapcar '* UV (mapcar '- Z O)))
                                         0 (* I OZ Q0)))
                    ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq I (expt OX 2) J (expt OY 2) K (expt OZ 2)
        NORM-XY (equal (setq X-O-Y (sqrt (+ I J)))
                       (setq X-Y (distance X Y)) (* X-Y Q0))
        NORM-Z (or 2D (and (equal (sqrt (+ I K)) (setq W (distance X Z)) (* W Q0))
                           (equal (sqrt (+ J K)) (setq W (distance Y Z)) (* W Q0))))
        BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (if (and NORM-XY NORM-Z)
      (progn
        (eval (append '(command "INSERT" NLOC-1 O "XYZ" OX OY (* OZ I) 0)
                      (if ATREQ WWAA-1)))
        (if OO-4 (INSERT** BLOC OX OY (* OZ I) 0)))
      (INSERT*))
  (command "SCP" "PR"
           "INSNAME" BLOC
           "EXPERT" EXPERT
           "ATTDIA" ATDIA
           "OSMODE" OSN
           "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))

(defun C:BLOQUEOK (/ Q BLOC I J K LK SA SS ULT CAPA ECO EXPERT AFL)
  (prompt "\nImprescindible para definir un bloque con atributos editables ")
  (prompt "justificados")
  (prompt "\nen su punto Medio, si SCP no es paralelo al SCU o el punto base ")
  (prompt "no es *0,0,0.")
```

```
  (while (not BLOC)
         (setq BLOC (getstring "\nIndique nombre de bloque o [?]: " T))
         (while (= (substr BLOC 1 1) " ")
                (setq BLOC (substr BLOC 2)))
         (if (> (setq I (strlen BLOC)) 0)
             (while (= (substr BLOC I 1) " ")
                    (setq I (1- I) BLOC (substr BLOC 1 I))))
         (setq BLOC (if (wcmatch BLOC (strcat ",*[" (chr 1) "-" (chr 31)
                                              "]*,*\"*,*`**,*`,*," "*/*,*["
                                              (chr 58) "-" (chr 62)
                                              "]*,*?`?*,*`??*," "*\\*,*``*,*|*,*["
                                              (chr 127) "-" (chr 160) "]*"))
                          (prompt "\nNombre bloque no válido.")
                          BLOC)))
  (setq Q T ECO (getvar "CMDECHO"))
  (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
  (setvar "CMDECHO" 0)
  (if (= BLOC "?")
      (command "BLOQUE" "?" (progn
                              (prompt "\nIndique bloque(s) a enumerar <*>: ")
                              (setvar "CMDECHO" 1) PAUSE))
      (if (or (not (tblsearch "BLOCK" BLOC))
              (= "Si" (progn
                        (initget "Si No")
                        (getkword (strcat "\nEl bloque \"" BLOC "\" ya existe. "
                                          "¿Desea volver a definirlo? [Sí/No] <N>: "))))))
          (progn
            (setq CAPA (getvar "CLAYER")
                  EXPERT (getvar "EXPERT")
                  AFL (getvar "AFLAGS")
                  I (initget 1) I (getpoint "\nPrecise punto base de inserción: ")
                  SA (ssget) SS (ssadd) ULT (entlast) J -1)
            (command "COPIA" SA "" "0,0,0" "")
            (while ULT (if (setq ULT (entnext ULT)) (ssadd ULT SS)))
            (while (setq J (1+ J) K (ssname SA J))
                   (setq LK (entget K))
                   (if (and (= (cdr (assoc 0 LK)) "ATTDEF")
                            (= (cdr (assoc 72 LK)) 4)
                            (= (logand (cdr (assoc 70 LK)) 2) 0))
                       (progn
                         (if Q (setq Q (alert
                                         (strcat "ATENCIÓN:\n\nEn los atributos "
                                         "editables seleccionados,\n\nel modo de "
                                         "justificación M se sustituye por MC"))))
                         (entmod (subst (cons 72 1)
                                        (cons 72 4)
                                        (subst (cons 74 2)
                                               (assoc 74 LK)
                                               LK))))))
            (command "CLAYER" "0"
                     "EXPERT" 2
                     "AFLAGS" 9
                     "ATRDEF" "" "BLOQUEOK" "" "" I "" ""
                     "BLOQUE" BLOC I (entlast) SA ""
                     "BORRA" SS ""
                     "AFLAGS" AFL
                     "EXPERT" EXPERT
                     "CLAYER" CAPA))))
  (command "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))

(defun ENTPRECEDING ()
  (setq A (entnext) K A)
  (while (not (equal K E)) (setq A K K (entnext K)))
  A)
```

```
(defun COMPR-OBLIC ()
  (setq EX (cdr (assoc 41 LE))
        EY (cdr (assoc 42 LE))
        EZ (cdr (assoc 43 LE)))
  (not (and (equal EX EY K0) (equal EY EZ K0) (equal EZ EX K0))))

(defun ATRIBS (E SS)
  (while (and E (or (not (setq LE (entget E) EZ (= (cdr (assoc 0 LE)) "ATTDEF")))
                    (= (logand (cdr (assoc 70 LE)) 2) 2)))
         (if SS
             (progn
               (if (and EZ (> EY 0) (wcmatch (cdr (assoc 2 LE)) "WWAA`-[12]"))
                   (progn (entdel E) (setq EY (1- EY))))
               (setq K (1+ K) E (ssname SA K)))
             (setq E (entnext E))))
  E)

(defun REST-VERIF ()
  (setq A (cdr (assoc 2 (entget E*))))
  (if (wcmatch A "VRF`**")
      (progn
        (setq LE (entget E)
              LE (subst (cons 2 (substr A 5)) (cons 2 A) LE)
              LE (subst (cons 70 (+ (cdr (assoc 70 LE)) 4))
                        (assoc 70 LE) LE))
        (entmod LE))))

(defun RESTAURA () (REST-VERIF) (if COMPROBLIC (REST-OBLIC)))

; VERSIÓ 22A++
(defun C:DESCOMPOK (/ K0 Q0 ECO BLOC E LE E* SA A K N NEXTE EX EY EZ COMPROBLIC)
  (setq K0 9.99994e-11 Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D (si hay atributos ")
  (prompt "con caracteres")
  (prompt "\noblicuos o no situados en su plano base, la orden DESCOMP no lo ")
  (prompt "resolverá bien).")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC (cdr (assoc 2 LE))
              BLOC (if (wcmatch BLOC "ATRIBATIP_#*")
                       (progn
                         (setq NEXTE E)
                         (while (or (/= (cdr (assoc 0 (setq E (ENTPRECEDING)
                                                            LE (entget E))))
                                        "INSERT")
                                    (if (wcmatch (cdr (assoc 2 LE))
                                                 "ATRIBATIP_#*")
                                        (setq NEXTE E))))
                         (cdr (assoc 2 LE)))
                       BLOC)
              ECO (getvar "CMDECHO"))
        (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
      ;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
        (setvar "CMDECHO" 0)
        (setq A (entnext E) K (= (last (trans (cdr (assoc 10 LE)) E 0)) 0))
        (command "DESCOMP" E ())
        (if (setq SA (ssget "P"))
            (progn
              (setq NEXTE (if NEXTE NEXTE (entnext E))
                    E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
                    COMPROBLIC (COMPR-OBLIC))
              (if (wcmatch BLOC (strcat "*_" N ",*_" N N))
                  (progn
                    (setq EZ (= (cdr (assoc 0 (entget (ssname SA 0)))) "INSERT")
                          K (if EZ 0 -1))
```

```
                    (while (setq K (1+ K) E (ssname SA K) E* (entnext E*))
                           (if (and (= K (if EZ 1 0))
                                    (= (cdr (assoc 2 (entget E))) "NO-BLOQUEOK"))
                               (entdel E)
                               (RESTAURA))))
                  (progn
                    (setq A (not (or (not A) (equal A NEXTE) COMPROBLIC
                                     (> (atoi (substr (getvar "ACADVER") 1 2)) 15)
                                     (and K (equal (cdr (assoc 210 LE))
                                                   '(0 0 1)))))
                          K 0 E (ssname SA K))
                    (if (wcmatch BLOC "*_AMB_ATRIBSTIPS")
                        (setq EX T EY 2)
                        (setq EY 0 LE (entget E*)
                              EX (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                      (= (cdr (assoc 2 LE)) "BLOQUEOK"))
                              EX (if EX EX (if A (AVIS)))))
                    (while (setq E (if (= K 0) (ATRIBS E T) E) E* (ATRIBS E* ()))
                           (if (and EX (wcmatch (cdr (assoc 2 LE)) "*BLOQUEOK"))
                               (progn (entdel E) (setq EX ()))
                               (RESTAURA))
                           (setq K (1+ K) E (ssname SA K)
                                 E* (entnext E*)))))))
        (setq K 0)
        (while (and (setq E NEXTE)
                    (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                    (wcmatch (setq COMPROBLIC (COMPR-OBLIC)
                                   BLOC (cdr (assoc 2 LE)))
                             (setq A (strcat "ATRIBATIP_" (itoa K) "_DE_*"))))
               (command "DESCOMP" E ())
               (setq NEXTE (entnext E)
                     E (ssname (ssget "P") 0)
                     E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
                     E* (if (wcmatch BLOC (strcat A "_" N)) (entnext E*) E*))
               (RESTAURA))
        (command "DESHACER" "F")
        (setvar "CMDECHO" ECO))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))

; VERSIÓ 22B++
(defun C:DESCOMPOK (/ K0 Q0 ECO BLOC E LE E* SA A K N NEXTE EX EY EZ COMPROBLIC)
  (setq K0 9.99994e-11
        Q0 0.001 N "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq N (strcat N "#")))
  (prompt "\nExplosione un bloque insertado con INS2D/INS3D (si hay atributos ")
  (prompt "con caracteres")
  (prompt "\noblicuos o no situados en su plano base, la orden DESCOMP no lo ")
  (prompt "resolverá bien).")
  (while (not (setq E (car (entsel)))))
  (setq LE (entget E))
  (if (= (cdr (assoc 0 LE)) "INSERT")
      (progn
        (setq BLOC (cdr (assoc 2 LE))
              BLOC (if (wcmatch BLOC "ATRIBATIP_#*")
                       (progn
                         (setq NEXTE E)
                         (while (or (/= (cdr (assoc 0 (setq E (ENTPRECEDING)
                                                            LE (entget E))))
                                        "INSERT")
                                    (if (wcmatch (cdr (assoc 2 LE))
                                                 "ATRIBATIP_#*")
                                        (setq NEXTE E))))
                         (cdr (assoc 2 LE)))
                       BLOC)
              ECO (getvar "CMDECHO"))
        (QUASI-SENSE-RASTRE) ; ALTERNATIVA: desactiva aquesta línia
      ;;; (SENSE-RASTRE)     ; i activa les que comencin amb ";;;".
```

```
        (setvar "CMDECHO" 0)
        (setq A (entnext E) K (= (last (trans (cdr (assoc 10 LE)) E 0)) 0))
        (command "DESCOMP" E ())
        (if (setq SA (ssget "P"))
            (progn
              (setq NEXTE (if NEXTE NEXTE (entnext E))
                    COMPROBLIC (COMPR-OBLIC)
                    A (not (or (not A) (equal A NEXTE) COMPROBLIC
                               (> (atoi (substr (getvar "ACADVER") 1 2)) 15)
                               (and K (equal (cdr (assoc 210 LE)) '(0 0 1)))))
                    K 0 E (ssname SA K)
                    E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
              (if (wcmatch BLOC (strcat "*_" N ",*_" N N ",*_AMB_ATRIBSTIPS"))
                  (setq EX T EY 2)
                  (setq EY 0 LE (entget E*)
                        EX (and (= (cdr (assoc 0 LE)) "ATTDEF")
                                (= (cdr (assoc 2 LE)) "BLOQUEOK"))
                        EX (if EX EX (if A (AVIS)))))
              (while (setq E (if (= K 0) (ATRIBS E T) E) E* (ATRIBS E* ()))
                     (if (and EX (wcmatch (cdr (assoc 2 LE)) "*BLOQUEOK"))
                         (progn (entdel E) (setq EX ()))
                         (RESTAURA))
                     (setq K (1+ K) E (ssname SA K) E* (entnext E*)))))
        (setq K 0)
        (while (and (setq E NEXTE)
                    (= (cdr (assoc 0 (setq K (1+ K) LE (entget E)))) "INSERT")
                    (wcmatch (setq COMPROBLIC (COMPR-OBLIC)
                                   BLOC (cdr (assoc 2 LE)))
                             (setq A (strcat "ATRIBATIP_" (itoa K) "_DE_*"))))
               (command "DESCOMP" E ())
               (setq NEXTE (entnext E)
                     E (ssname (ssget "P") 0)
                     E* (cdr (assoc -2 (tblsearch "BLOCK" BLOC)))
                     E* (if (wcmatch BLOC (strcat A "_" N)) (entnext E*) E*))
               (RESTAURA))
        (command "DESHACER" "F")
        (setvar "CMDECHO" ECO))
      (prompt "\n\nEsto no es ninguna inserción de bloque."))
  (princ))
```

La proposta que farem ara no comporta el més mínim avantatge per a la VERSIÓ 22++ i sols és un formalisme que repercuteix sobre el nom dels blocs derivats genèrics, però ens permetem aquesta digressió de cara a una futura VERSIÓ 23++ on sí que introduirem una notable simplificació en l'arbre format pels genèrics teòricament disponibles que pengen de **BLOC:** els noms que en la nova versió hauran de rebre els genèrics **BLOC**** que responguin a la situació **NORM-XY** i els genèrics **BLOC*** (**NORM-Z**) s'entendran millor si abans apliquem aquests criteris de denominació a la present.

De primer hem d'entendre que els genèrics **BLOC**** aplicables en condicions **NORM-XY** estan associats a prismes oblics de base quadrada, i hem convingut d'assignar-los noms del tipus *B_397643_859* (seguim amb l'exemple del **BLOC** de nom *B* i **Q0 = 0,001**) perquè estan constituïts per una inserció de *B_397643_SENSE_ATRIBS* i pels atributs resultants d'explosionar una inserció de *ATRIBS_DE_B_397643* (**45° − γ_1 = 397**, en una graduació en què **180°** són **999**, i cos **β_1 = 0,643**), insercions fetes amb els mateixos paràmetres: cos **β_2 = 0,859** (amb **β_1** i **β_2** la relació d'escales $\frac{E_y}{E_x}$ queda determinada) i cos **α_1 = 1,000**. És precisament la singularitat d'aquesta última dada allò que ens va decidir a no incorporar-ne la part decimal: més per l'estranya discontinuïtat dels noms per a valors creixents de **α_1** (com ara *B_397643_859000*, *B_397643_859999*, *B_397643_859998*, etc.) que no pas pel valor en si.

Tres quarts del mateix passa amb els genèrics **BLOC*** aplicables quan tenim **NORM-Z** i associats al cub ortonormal de **BLOC**, als quals hem convingut d'assignar-los noms del tipus *B_866* perquè estan constituïts per una inserció de *B_SENSE_ATRIBS* i pels atributs resultants d'explosionar una inserció de *ATRIBS_DE_B* (com que l'angle **γ_1**

no està definit per a paral·lelepípedes rectes i $\cos\beta_1 = 1{,}000$, no cal recórrer a un tàndem de genèrics de primer grau, sinó directament al de substituts de **BLOC**), insercions fetes amb els mateixos paràmetres: $\cos\beta_2 = 1{,}000$ (amb $\beta_1 = \beta_2 = 0°$ la relació d'escales $\frac{E_y}{E_x}$ pot tenir qualsevol valor i passarà a la inserció final) i $\cos\alpha_1 = 0{,}866$. La diferència és que aquí no té sentit parlar de discontinuïtat dels noms per a valors creixents de β_2 (com ara *B_000866*, *B_999866*, *B_998866*), perquè la continuïtat es produeix en totes les direccions, si en considerem una, la discontinuïtat quan a la denominació és insalvable de tota manera (el pas seria de *B_000866* a *B_397999_999866*, *B_397998_998866*, etc., posem per cas), i si ara ens entestem a afegir els tres zeros és per poder-los treure més endavant. No és pas una broma de mal gust: quan tornem a prescindir de l'additament que ara adoptem no ho farem de forma gratuïta sinó en el marc d'una operació més amplia i perquè la designació que ara convenim a donar-li a **BLOC*** ens ajudarà a copsar-ne l'abast.

Per tirar endavant la proposta i generalitzar els criteris d'atribució de noms, evitant les excepcions, anomenant *B_397643_859000* el que fins ara era *B_397643_859* i *B_000866* el que fins ara era *B_866*, només hauríem de canviar:

- A **REDEF-BLOC*S**:
  - on hi posa
    (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat BLOC "_" M "*")) ...)
    hauria de posar-hi
    (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat BLOC "_" M M "*")) ...)
  - on hi posa
    (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat BLOC-2 "_" M)) ...)
    hauria de posar-hi
    (if (wcmatch (setq BLOC* (cdr (assoc 2 LB))) (strcat BLOC-2 "_" M M)) ...)

- A **INSERT***:
  - on hi posa

```
(if NORM-Z
    (setq BLOC* (strcat BLOC "_"))
    (progn
      ..........................................
      (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*
            BLOC* (strcat BLOC* "_" (FIX+ (/ OXY** OX* Q0))))))
(if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
```

    hauria de posar-hi

```
(if NORM-Z
    (setq BLOC* BLOC OXY** OX*)
    (progn
      ..........................................
      (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*)))
(setq BLOC* (strcat BLOC* "_" (FIX+ (/ OXY** OX* Q0)) (FIX+ (/ OX** OX* Q0))))
```

  - A **C:DESCOMPOK** (VERSIÓ 22A++):
    - on hi posa
      (if (wcmatch BLOC (strcat "*_" N ",*_" N N)) ...)
      hauria de posar-hi
      (if (wcmatch BLOC (strcat "*_" N N)) ...)
    - on hi posa
      (setq ... E* (if (wcmatch BLOC (strcat A "_" N)) (entnext E*) E*))
      hauria de posar-hi
      (setq ... E* (if (wcmatch BLOC (strcat A "_" N N)) (entnext E*) E*))

  - A **C:DESCOMPOK** (VERSIÓ 22B++):
    - on hi posa
      (if (wcmatch BLOC (strcat "*_" N ",*_" N N ",*_AMB_ATRIBSTIPS")) ...)
      hauria de posar-hi
      (if (wcmatch BLOC (strcat "*_" N N ",*_AMB_ATRIBSTIPS")) ...)
    - on hi posa
      (setq ... E* (if (wcmatch BLOC (strcat A "_" N)) (entnext E*) E*))
      hauria de posar-hi
      (setq ... E* (if (wcmatch BLOC (strcat A "_" N N)) (entnext E*) E*))

Amb només aquests canvis el sistema encara grinyolaria, per les discontinuïtats que hem esmentat (*B_397643_859000*, *B_397643_859999*, *B_397643_859998*... d'una banda i *B_000866*, *B_999866*, *B_998866*... de l'altra). Per tal de superar-les, no caldria arribar a substituir la part decimal de la funció cosinus per la del sinus dels angles β_1, β_2 i α_1, i n'hi hauria prou a adoptar la diferència a **1** del cosinus. El procediment seria tan senzill com anar una mica més enllà en la modificació de la funció **INSERT***, on en comptes de canviar el fragment

```
(if NORM-Z
    (setq BLOC* (strcat BLOC "_"))
    (progn
      .....................................................
      (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                          (FIX+ (/ OXY** OXY* Q0))) ...)
      ...................................................
      (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*
            BLOC* (strcat BLOC* "_" (FIX+ (/ OXY** OX* Q0))))))
(if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
```

tal com havíem indicat, el deixaríem així:

```
(if NORM-Z
    (setq BLOC* BLOC OXY** OX*)
    (progn
      .....................................................
      (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                          (FIX+ (/ (- 1 (/ OXY** OXY*)) Q0))) ...)
      ...............................................
      (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*)))
(setq BLOC* (strcat BLOC* "_" (FIX+ (/ (- 1 (/ OXY** OX*)) Q0))
                              (FIX+ (/ (- 1 (/ OX** OX*)) Q0))))
```

Tanmateix, volem aprofitar l'avinentesa per revisar una decisió presa des d'un bon començament (NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS) i que després hem mantingut i refermat (ENCAIX 3D DE BLOCS AMB ATRIBUTS), malgrat una consistència pràctica més aviat feble: el recurs a $\cos\beta_1$, $\cos\beta_2$ i $\cos\alpha_1$ en lloc dels angles β_1, β_2 i α_1; de fet, només l'angle γ_1 s'ha utilitzat directament en la composició del nom dels blocs genèrics. No era una decisió arbitrària i, de fet, en l'últim dels capítols citats hi havia una justificació d'aquest *modus operandi*, referida a β_1 però que aplicàvem també als demés:

> "... Quant a l'angle β_1, igual que en el capítol NOUS RECURSOS: ENCAIX 2D DE BLOCS SENSE ATRIBUTS, no figurarà aquest valor en el nom de **BLOC*** sinó la part decimal de $\cos\beta_1$. La substitució de l'angle pel cosinus no suposa perdre precisió sinó just al contrari, perquè ens interessa parametritzar els resultats a partir d'aquesta funció trigonomètrica (els valors de la qual obtenim de primera mà i ens proporcionen directament el factor d'escala amb què s'insereix **BLOC**) i no a partir de l'angle..."

Efectivament, almenys en el cas de $\cos\beta_1$ i $\cos\alpha_1$ estava clar que aquests valors representaven la relació d'escales $\frac{E_y}{E_x}$ en la inserció de **BLOC*** (cas **NORM-Z**, en què era la inserció final) o de **BLOC-1*** i **BLOC-2*** (resta de casos), i en la de **BLOC**** (inserció final quan era **(not NORM-Z)**). Tenia sentit doncs, la utilització directa d'aquests valors per identificar els genèrics, en comptes de fer-ho amb uns altres paràmetres calculats a partir d'ells, perquè així evitàvem la irrupció d'errors de precisió que poguéssin allunyar els genèrics (disponibles o encara per crear) dels seus dominis geomètrics d'aplicació: que dos paral·lelepípedes d'encaix gairebé coincidents haguéssin de recórrer a genèrics diferents (tot i que en un lloc o un altre s'ha de situar la frontera) o que un mateix genèric fos reclamat des de dos encaixos tan diferents com per donar lloc a resultats poc satisfactoris (tot i que la discretització que deriva del valor **Q0** adoptat porta implícita una tolerància). De tota manera, si som realistes i considerem els genèrics teòricament disponibles que podrien penjar de **BLOC** (hauríem de comptar amb $\frac{1}{Q0^2}$ genèrics de primer grau, entre **BLOC***s i tàndems **BLOC-1***/**BLOC-2***, i amb $\frac{1}{Q0^4}$ genèrics de segon grau **BLOC****, que l'anunciada VERSIÓ 23++ aconseguirà reduir a "només" $\frac{1}{Q0^3}$), cal admetre que: o bé escollim uns valors **Q0** no massa petits i acceptem una tolerància important, o bé escollim valors prou petits com perquè la tolerància sigui més assumible però

adoptem limitacions en la geometria dels paral·lelepípedes d'encaix, modularitzant l'espai de cara al posicionament dels punts **O**, **X**, **Y** i **Z**, o aplicant restriccions específiques de l'activitat des de la qual utilitzem ***INS2D***/***INS3D***.

De tota manera, presentar la disjuntiva entre l'angle i el seu cosinus com un tema amb transcendència pràctica en el terreny de la precisió (el segon valor com una dada de "primera mà", en contraposició al primer, al qual hi arribem indirectament i corrent el perill de perdre exactitud pel camí) és passar per alt com obteniem aquestes dades de "primera mà": la construcció de Ritz, implementada en **CALCULA**, on es juga amb funcions predefinides com **polar**, **angle**, **inters** i **distance**, no n'és precisament el paradigma. També és oblidar-se del que hem tingut ocasió de veure just abans d'oferir el codi complet de la VERSIÓ 22++, quan en cercar l'origen d'unes anomalies que es produïen a **REDEF-BLOC*S**, les trobàvem a **INSPARCIAL** i, més exactament, en una definició de **COMPROBLIC** que desconeixia les intimitats de la computació en AutoCAD: allà hem constatat que la precisió de càlcul arriba força més enllà del topall voluntàriament instal·lat en adoptar un valor **Q0** raonable.

D'altra banda, la consecució d'una precisió homogènia (homogènia respecte a quin paràmetre?) no és pas el problema principal, sinó mirar de filar prim en l'entorn d'uns valors crítics on les variacions quantitatives comporten canvis qualitatius: ens referim als entorns de $\beta_1 = 0°$ (on, de ser **NORM-Z** i assolir-se l'encaix amb un **BLOC*** *B_000333*, es passa a **(not NORM-Z)** i necessitar un **BLOC**** *B_397001_001333*) i de $\alpha_1 = 0°$ (on, de ser **NORM-XY** i assolir-se l'encaix amb un **BLOC**** *B_397555_342000* es passa a **(not NORM-XY)** i necessitar un **BLOC**** *B_397556_342001*), mal tractats per les funcions trigonomètriques $\cos\beta_1$ i $\cos\alpha_1$, que en aquest punt tenen derivada zero i que, amb les variacions angulars realitzades i reflectides directament en el nom dels genèrics, no registrarien cap canvi; això per no citar el pas de les condicions simultànies $\beta_1 = 0°$ i $\alpha_1 = 0°$ a unes de pròximes on, de ser **NORM-Z** i **NORM-XY** i assolir-se l'encaix inserint directament *B_SENSE_ATRIBS* i *ATRIBS_DE_B*, gairebé sense adonar-nos ens plantem en qualssevol de les altres tipologies. Són consideracions que, unides a una major facilitat per fitar mètricament els errors, com veurem de seguida, ens han empès a donar el pas per canviar de bàndol.

Com que l'autor ja ha confessat ser poca cosa més que un voluntariós aficionat, li estalviarà al lector haver de suportar penosos arguments exculpatoris en el sentit de no dominar la Teoria d'Errors i no haver introduït en ***INS2D***/***INS3D*** cap criteri coherent per fer compatibles els errors derivats de l'aplicació de la tolerància **Q0**, de manera indiscriminada, en àmbits diferents. Això no exclou que qualsevol intervenció correctora per part de lectors tan voluntariosos com experts en tal matèria serà ben rebuda, i només els demanaria que donessin lliure difusió a la seva aportació, en justa correspondència al lliure accés que han tingut al text que estan llegint. Calia dir això abans d'embrancar-se en una reflexió que només aspira a pal·liar els efectes més negatius de l'omissió denunciada, en relació a l'objecte de la qual no pot pretendre ser altra cosa que un pobre succedani.

REDEF-BLOC*S i **C:DESCOMPOK** mantindran els canvis relatius als perfils dels noms. I tornant a **INSERT***, al punt exacte on havíem començat l'adaptació del codi als nous criteris de denominació, als quals s'integra ara el canvi del cosinus per l'angle, veureu que la nova funció **ANG** està proveïda d'uns dispositius per evitar que pugui aparèixer un angle de **1000** graus mil·lessimals (usats quan **Q0 = 0,001**, com encara seguim suposant), equivalent a **90°**, o un de **000** quan no sigui $\alpha_1 = 0°$ (**NORM-XY**) ni $\beta_1 = 0°$ (**NORM-Z**), i ens engeguin en orris el muntatge. Tot això a més dels canvis d'estructura suggerits abans en unificar-se els noms dels **BLOC****s, sense distinció entre el cas general i **NORM-XY** pel que fa a nombre de caràcters, i compartir amb **BLOC*** el segon sufix. Entre aquesta unificació i el relleu de cosinus per angle, serà convenient ampliar **CALCULA** amb l'argument **OV**** (determinació de **OY**** i **OZ****):

```
(defun CALCULA (U V OV AGUT U* U** V** OU* OU** OV**)
  (setq A (polar O (- (angle O V) PI/2) OV)
        W (if AGUT (angle A U) (angle U A))
        B (mapcar '/ (mapcar '+ A U) '(2 2 2))
        W (angle O (polar B W (distance O B))))
  (set U* (inters U (polar U W 1) O B ()))
  (set U** (inters O (polar O (+ W PI/2) 1) U (eval U*) ()))
  (set V** (inters V (polar V W 1) O (eval U**) ()))
  (set OU* (distance O (eval U*)))
  (set OU** (distance O (eval U**)))
  (set OV** (distance O (eval V**))))
```

```
(defun ANG (S C NORM / A)
  (setq A (/ (atan S C) PI/2 Q0)
        A (if (< (- (/ 1 Q0) A) 1)
              (1- (/ 1 Q0))
              (if (< A 1) (if NORM 0 1) A)))
  (FIX+ A))

(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** OY** Z-XY* XY XY* XY** OXY* OXY**
                  OZ** ABLOC*)
  ..........................................................
  (if NORM-XY
      (setq X* X X** X
            Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX OY** 0
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
      (progn
        (CALCULA X Y OY (< X-Y X-O-Y) 'X* 'X** 'Y** 'OX* 'OX** 'OY**) ...))
  (if NORM-Z
      (setq BLOC* BLOC OXY** OX* Z** Z OZ** OZ)
      (progn
        ....................................
        (CALCULA XY Z (distance O Z) (> (car Z) 0) 'XY* 'XY** 'Z** 'OXY* 'OXY**
                                                   'OZ**)
        (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                            (ANG OZ** OXY** ())) ...)
        ....................................................................))
  (setq BLOC* (strcat BLOC* "_"
                      (ANG (* OZ** (distance Z Z**)) (expt OXY** 2) NORM-Z)
                      (ANG OY** OX** NORM-XY)))
  ..........................................................................)
```

Cal no confondre aquest dispositiu amb el paràmetre adoptat per representar en els noms **BLOC-1*** i **BLOC-2*** l'angle $45° - \gamma_1$ (capítol ENCAIX 3D DE BLOCS AMB ATRIBUTS), que mesuràvem en graus **(1- (/ 1 Q0))**essimals perquè el rang era $0° \le 45° - \gamma_1 \le 180°$ i no volíem ultrapassar les tres xifres: si **Q0 = 0,001** dos rectes se subdivideixen en **999** graus. Per contra, $0° < \alpha_1 < 90°$ i $0° < \beta_1 < 90°$ perquè els valors límit mai no poden resultar de la construcció de Ritz (de fet, quan es dóna **NORM-XY** o **NORM-Z** el procés es singularitza i no s'accedeix a la funció **CALCULA**), i malgrat això es mesuren en graus **(/ 1 Q0)**essimals: si **Q0 = 0,001** un angle recte se subdivideix en **1000** graus. És també per aquesta raó (l'altra raó és que γ_1 s'avalua directament, mentre que β_1, β_2 i α_1 ho fan a partir de funcions trigonomètriques) que ens hem estimat més no forçar les coses per unificar la concurrència a **FIX+** (que els tres últims angles fan mitjançant **ANG**).

És legítim de preguntar-se per què hem hagut de recórrer a un artifici tan barroer com ara la conversió expeditiva d'angles gairebé rectes, avaluats com a **1000** graus mil·lessimals, a **999** (per tal de no sobrepassar el nombre de dígits establert), i la d'angles gairebé nuls però no derivats de les singularitats **NORM-XY** o **NORM-Z**, avaluats com a **000** graus mil·lessimals, a **001** (per tal de no coincidir amb valors reservats per a les singularitats esmentades): que potser no podíem haver detectat aquestes situacions extremes en l'origen (en el procés d'entrada dels punts **O**, **(X)**, **(Y)** i **(Z)**) i filtrar-les allà mateix? Doncs rotundament no, perquè una cosa és l'angle **(X)–O–(Y)** i el format per **(Z)** i el pla **(X)–O–(Y)** (serà millor tornar-nos a acostumar a la notació utilitzada en el penúltim capítol, en comptes de la més immediata **X**, **Y**, **Z** i **X–O–Y**), una altra els punts **(X′)**, **(Y′)**, **(Z′)** i **(XY′)**, una altra de diferent els punts $(XY')'$ i $(Z')'$, i encara una altra els angles $\alpha_1 = (X')''–O–(X')'$ i $\beta_1 = (XY')''–O–(XY')'$: en els dos primers angles sí que podem aplicar el control que veurem tot seguit, de primer per evitar que **O**, **(X)** i **(Y)** estiguin alineats o que siguin coplanaris amb **(Z)** i després per determinar si es poden qualificar com a casos **NORM-XY** o **NORM-Z**, pero d'aquí als dos últims angles hi ha tot un procés complex marcat per l'encadenament de dues construccions de Ritz, procés en què intervé la relació de longituds $\frac{O–(Y)}{O–(X)}$ i $\frac{O–(Z')}{O–(XY')}$ de manera tan decisiva com els

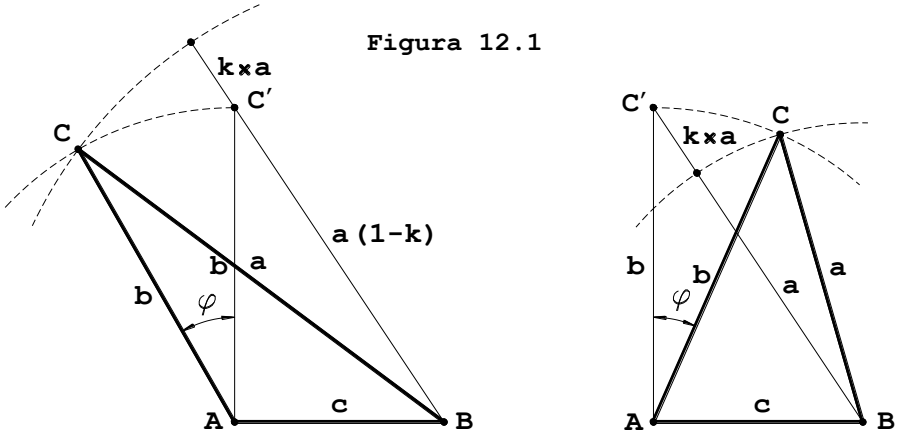
angles **(X)–O–(Y)** i **(XY′)–O–(Z′)**. Si n'analitzem les implicacions, veurem que fins i tot en el procediment per a la detecció d'aquestes singularitats ens hem complicat la vida gratuïtament: malgrat les esmenes introduïdes a la VERSIÓ 22++, convertint les toleràncies de **equal** en relatives, la caracterització de **NORM-XY** i de **NORM-Z**

```
(setq I (expt OX 2) J (expt OY 2) K (expt OZ 2)
      NORM-XY (equal (setq X-O-Y (sqrt (+ I J)))
                     (setq X-Y (distance X Y)) (* X-Y Q0))
      NORM-Z (or 2D
                 (and (equal (sqrt (+ I K)) (setq W (distance X Z)) (* W Q0))
                      (equal (sqrt (+ J K)) (setq W (distance Y Z)) (* W Q0))))
      BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
```

està coixa i tindrà un comportament irregular, perquè només la fem dependre de les longituds **O–(X)**, **O–(Y)**, **O–(Z)**, **(X)–(Y)**, **(X)–(Z)** i **(Y)–(Z)** primàriament, quan hauria de dependre també de les relacions $\frac{\text{O–(Y)}}{\text{O–(X)}}$, $\frac{\text{O–(Z)}}{\text{O–(X)}}$ i $\frac{\text{O–(Z)}}{\text{O–(Y)}}$. Veiem-ho.

Simplifiquem la notació i considerem el triangle **A–B–C**, aproximadament rectangle, de la Figura 12.1. Suposant que els costats **b** i **c** tenen una longitud inamovible, i que només **a** pot variar la seva longitud entre els límits $\mathbf{a\,(1-k) \le a \le a\,(1+k)}$, els dibuixos a l'esquerra i a la dreta representen els casos extrems de triangles d'angle **B–A–C** obtús/agut que, escurçant/allargant **a** dins dels límits previstos, poden convertir-se en el triangle rectangle autèntic **A–B–C′**, i volem avaluar els angles φ_1 i φ_2, amb vèrtex en **A**, definits pels increments de **a** (angles **C–A–C′**).

Figura 12.1



$$\mathbf{a^2\,(1 \mp k)^2 = b^2 + c^2 \rightarrow a^2 = \frac{b^2 + c^2}{(1 \mp k)^2}}$$

Si, per simplificar, situem **A** en el origen de coordenades i escalem el conjunt fins que sigui $\mathbf{a = 1}$, les coordenades de **A** seran **0,0**, les de **B** seran **c,0** i les de **C′** seran **0,b**. Per localitzar el punt **C** partirem d'aquestes dues equacions:

$$\mathbf{C_x^2 + C_y^2 = b^2}$$

$$\mathbf{(C_x - B_x)^2 + (C_y - B_y)^2 = (C_x - c)^2 + C_y^2 = 1}$$

$$\mathbf{C_x^2 + c^2 - 2\,c\,C_x + C_y^2 = 1}$$

$$\mathbf{C_x^2 + C_y^2 = 1 - c^2 + 2\,c\,C_x + C_y^2 = b^2}$$

$$\mathbf{C_x = \frac{b^2 + c^2 - 1}{2\,c} = \frac{(1 \mp k)^2}{2\,c}}$$

$$\mathbf{C_y = \sqrt{b^2 - C_x^2} = \sqrt{b^2 - \frac{\left[(1 \mp k)^2 - 1\right]^2}{4\,c^2}} = \frac{\sqrt{4b^2c^2 - \left[(1 \mp k)^2 - 1\right]^2}}{2\,c}}$$

$$\tan \mathbf{\varphi} = \mathbf{\frac{C_x}{C_y} = \frac{(1 \mp k)^2 - 1}{\sqrt{4b^2c^2 - \left[(1 \mp k)^2 - 1\right]^2}}}$$

Per a cada valor **k**, el mínim tan φ (i el mínim φ) correspondrà al màxim de $\mathbf{b \times c}$. Si considerem tots els triangles rectangles d'hipotenusa $\mathbf{a = 1}$, els catets **b** i **c** representarien respectivament el cosinus i el sinus de l'angle $\mathbf{A{-}C'{-}B}$. Derivant respecte aquest angle la funció $\mathbf{f\,(A{-}C'{-}B) = \cos\,(A{-}C'{-}B) \times \sin\,(A{-}C'{-}B) = b \times c}$ i igualant a zero,

$$\mathbf{f'\,(A{-}C'{-}B) = -\sin\,(A{-}C'{-}B) \times \sin\,(A{-}C'{-}B) + \cos\,(A{-}C'{-}B) \times \cos\,(A{-}C'{-}B) = 0 \rightarrow}$$

$$\mathbf{\rightarrow \sin^2\,(A{-}C'{-}B) = \cos^2\,(A{-}C'{-}B)}$$

Entre $\mathbf{0°}$ i $\mathbf{90°}$, hi ha un màxim per a $\mathbf{A{-}C'{-}B = 45°}$, en què $\mathbf{b = c = 0{,}7071}$. Per a valors petits de **k**, el màxim de $\mathbf{b \times c}$ també es donarà amb valors $\mathbf{A{-}C{-}B \approx 45° \rightarrow}$ $\mathbf{\rightarrow b \approx c \approx 0{,}7071}$ i, com hem dit, correspondran als mínims angles $\mathbf{C{-}A{-}C'}$. Vejam, doncs, quins són aquests angles quan $\mathbf{k = 0{,}001}$, valor de referència de **Q0** adoptat:

$\mathbf{\varphi = \arctan \dfrac{(1 \mp k)^2 - 1}{\sqrt{1 - \left[(1 \mp k)^2 - 1\right]^2}}}$. Així doncs, amb $\mathbf{b \approx c \approx 0{,}7071}$, quan $\mathbf{a = 0{,}999}$ serà $\mathbf{\varphi_1 = -0{,}001999}$ radiants ($\mathbf{-0{,}114534°}$), i quan $\mathbf{a = 1{,}001}$ serà $\mathbf{\varphi_2 = 0{,}002001}$ radiants ($\mathbf{0{,}114649°}$). I, com que ja havíem decidit expressar els angles α_1 i β_1 en graus mil·lessimals, posem que $\mathbf{\varphi_1 = -1{,}27260}$ i $\mathbf{\varphi_2 = 1{,}27388}$. Però no en deturem en $\mathbf{\dfrac{b}{c} = 1}$ i seguim aplicant l'expressió $\mathbf{\varphi = \dfrac{2000}{\pi} \times \arctan \dfrac{(1 \mp k)^2 - 1}{\sqrt{4b^2c^2 - \left[(1 \mp k)^2 - 1\right]^2}}}$ a valors creixents de $\mathbf{\dfrac{b}{c}}$ per veure'n l'evolució:

$\mathbf{\dfrac{b}{c} = 1 \rightarrow \varphi_1 = -1{,}27260 \quad \varphi_2 = 1{,}27388}$

$\mathbf{\dfrac{b}{c} = 2 \rightarrow \varphi_1 = -1{,}59076 \quad \varphi_2 = 1{,}59235}$

$\mathbf{\dfrac{b}{c} = 5 \rightarrow \varphi_1 = -3{,}30878 \quad \varphi_2 = 3{,}31209}$

$\mathbf{\dfrac{b}{c} = 10 \rightarrow \varphi_1 = -6{,}42675 \quad \varphi_2 = 6{,}43318}$

$\mathbf{\dfrac{b}{c} = 20 \rightarrow \varphi_1 = -12{,}7587 \quad \varphi_2 = 12{,}7715}$

$\mathbf{\dfrac{b}{c} = 50 \rightarrow \varphi_1 = -31{,}8411 \quad \varphi_2 = 318730}$

No és gens probable que a la pràctica se'ns presentin aplicacions en què l'encaix s'hagi de fer en paral·lelograms on la relació entre dues arestes sigui superior a $\mathbf{50}$, però només que vulguem estar preparats per a $\mathbf{\dfrac{b}{c} = 50}$ ja tindrem un problema addicional per resoldre: partíem d'una tolerància lineal $\mathbf{Q0 = \pm\,0{,}001}$ aplicable a la hipotenusa d'un triangle rectangle, hem volgut saber quina tolerància angular comportava això i ens hem quedat amb un pam de nas, perquè aquesta encara depenia de la relació entre la longitud dels catets; únicament quan aquests eren iguals la tolerància angular ($\mathbf{\mp 1{,}273}$ graus mil·lessimals, si ens quedem amb un valor mitjà) era del mateix ordre que la utilitzada en el control de la independència lineal de **O**, **X**, **Y** i **Z** ($\mathbf{\pm\,0{,}001}$ com a intèrval de variació del sinus en $\mathbf{\sin 0° = \sin 180° = 0}$, del cosinus en $\mathbf{\cos 90° = \cos 270° = 0}$, que es corresponia a l'intèrval pràcticament idèntic de $\mathbf{\pm\,0{,}001001}$ radiants, equivalent a $\mathbf{\pm\,0{,}637256}$ graus mil·lessimals). Però a mesura que $\mathbf{\dfrac{b}{c}}$ augmenta la tolerància angular també ho fa, gairebé linealment (la funció $\mathbf{0{,}624\,\dfrac{b}{c} + 0{,}65}$, ajustada a $\mathbf{\dfrac{b}{c} = 1}$ i a $\mathbf{\dfrac{b}{c} = 50}$, presenta una desviació màxima de $\mathbf{0{,}45}$). En base a aquesta linealitat aproximada, podríem crear una funció **RECT** que reproduís la penúltima fòrmula (φ en radians), definís d'entrada la constant **AQ0** fent **(setq AQ0 (RECT 1.0))** i calculés també **(RECT (/ OY OX))**, **(RECT (/ OZ OX))** i **(RECT (/ OZ OY))**, expressions on hem suposat que $\mathbf{OX \leq OY \leq OZ}$. D'aquesta manera aconseguiríem treballar amb una tolerància angular pràcticament constant, partint en cada cas d'una tolerància lineal diferent, atès que des d'una tolerància lineal

relativa però constant no és possible treballar amb una tolerància angular única: el procediment consistiria a reduir la tolerància lineal **Q0 = ± 0,001** de base en la proporció **(/ AQ0 (RECT (/ B C)))** pròpia de cada relació $\frac{b}{c}$. Aquí teniu el codi:

```
(defun RECT (B//C / K B/C2 B2 C2)
  (setq K (1- (expt (1+ Q0) 2))
        B/C2 (expt B//C 2)
        B2 (/ B/C2 (1+ B/C2))
        C2 (/ 1 (1+ B/C2)))
  (atan K (sqrt (- (* 4 B2 C2) (expt K 2)))))

(defun INSERTOK (2D / K0 Q0 AQ0 ... E* I J K M N W NORM-XY NORM-Z OO-1 OO-2 OO-4)
  (setq K0 9.99994e-11
        Q0 0.001 M "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq M (strcat M "#")))
  (setq AQ0 (RECT 1.0))
        2*PI (* PI 2) PI/2 (/ PI 2)
        V>15 (> (atoi (substr (getvar "ACADVER") 1 2)) 15) ...)
  (while W
    ..............................................
    (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))
          I (distance '(0 0 0) UV)
          Z (if 2D (mapcar '+ O UV) Z)
          OZ (if 2D OX OZ)
          W (if (<= I (* OX OY Q0))
                " e Y están alineados."
                (if (and (not 2D)
                         (equal (apply '(lambda (X Y Z) (+ X Y Z))
                                       (mapcar '* UV (mapcar '- Z O)))
                                0 (* I OZ Q0)))
                    ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq I (expt OX 2) J (expt OY 2) K (expt OZ 2)
        NORM-XY (equal (setq X-O-Y (sqrt (+ I J)))
                       (setq X-Y (distance X Y))
                       (* X-Y Q0 (/ AQ0 (RECT (/ (max OX OY) (min OX OY))))))
        NORM-Z (or 2D
                   (and (equal (sqrt (+ I K))
                               (setq W (distance X Z))
                               (* W Q0 (/ AQ0 (RECT (/ (max OX OZ)
                                                       (min OX OZ))))))
                        (equal (sqrt (+ J K))
                               (setq W (distance Y Z))
                               (* W Q0 (/ AQ0 (RECT (/ (max OY OZ)
                                                       (min OY OZ)))))))))
        BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))) ..............)
```

Ara bé: si l'objectiu és utilitzar una tolerància angular constant (o, si més no, constant per a cada control, procurant a més que les diferents constants siguin d'un mateix ordre de magnitud), ¿per què ens compliquem la vida entestant-nos a partir d'una definició lineal?. El mal ja ve de lluny, perquè per a la detecció dels casos **NORM-XY** i **NORM-Z** no calia haver-se embolicat amb un algorisme diferent (l'acompliment de Pitàgores en els triangles **(X)–O–(Y)**, **(X)–O–(Z)** i **(Y)–O–(Z)**), sinó que haguéssim pogut explotar dos valors que ens havíem molestat a calcular per al control de la independència lineal dels punts **O**, **X**, **Y** i **Z**: el producte vectorial **UV** de **O–(X)** i **O–(Y)**), i el producte escalar de **UV** per **O–(Z)**. Si en aquest control havíem de vigilar que no fossin **0**, **NORM-XY** i **NORM-Z** s'esdevindran quan valguin **1**, respectivament, i tot plegat quedaria reduït a

```
  (while W
    ..............................................
    (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))
          I (distance '(0 0 0) UV)
          J (/ I (* OX OY))
          Z (if 2D (mapcar '+ O UV) Z)
          OZ (if 2D OX OZ)
```

```
        K (if 2D 1 (/ (apply '(lambda (X Y Z) (+ X Y Z))
                             (mapcar '* UV (mapcar '- Z O)))
                      (* I OZ)))
        W (if (<= J Q0)
              " e Y están alineados."
              (if (and (not 2D) (equal K 0 Q0))
                  ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq NORM-XY (equal J 1 Q0)
        NORM-Z (or (equal K 1 Q0) (equal K -1 Q0))
        X-Y (expt (distance X Y) 2)
        X-O-Y (+ (expt OX 2) (expt OY 2))
        BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
```

on **X-Y** i **X-O-Y** ja només els necessitem per assignar el valor lògic **(< X-Y X-O-Y)** a l'argument **AGUT** de la funció **CALCULA**, el primer cop que hi accedim des de **INSERT***.

No hi ha color quant a simplicitat, però també aquí tenim un problema. Si anomenem α l'angle **(X)–O–(Y)** i β l'angle que forma **O–(Z)** amb el pla **(X)–O–(Y)** (és un recurs provisional que no té res a veure amb els α, α_1, α_2, β, β_1 i β_2 vistos), **J** és $\sin\alpha$ i **K** és $\cos\beta$. En el primer control apliquem les condicions $0 \le \sin\alpha \le 0{,}001$ i $-0{,}001 \le \cos\beta \le 0{,}001$, equivalents a $0 \le \alpha \le 0{,}001$ o $3{,}1416 \le \alpha \le 3{,}1426$ radiants, i a $1{,}5698 \le \beta \le 1{,}5718$ o $4{,}7114 \le \beta \le 4{,}7134$ radiants, respectivament, la qual cosa indica que la tolerància aplicada a $\alpha = 0$ o $\alpha = \pi$, i a $\beta = \frac{1}{2}\pi$ o $\beta = \frac{3}{2}\pi$, és pràcticament igual a la corresponent a les funcions trigonomètriques $\sin\alpha$ i $\cos\beta$ aplicades a aquests valors (en realitat, ja hem dit més amunt que el valor exacte és $\pm 0{,}001001$ radiants o $\pm 0{,}637256$ graus mil·lessimals). Però quan ens situem a $\alpha = \frac{1}{2}\pi$ i a $\beta = 0$ o $\beta = \pi$, les coses canvien perquè els intèrvals $0{,}999 \le \sin\alpha$ i $0{,}999 \le \cos\beta$ o $\cos\beta \le -0{,}999$, es corresponen a $1{,}5261 \le \alpha \le 1{,}6155$ radiants, i a $-0{,}0447 \le \beta \le 0{,}0447$ o $3{,}0969 \le \beta \le 3{,}1863$ radiants, cosa que revela que la tolerància angular és gairebé **45** vegades més gran que la trigonomètrica ($\pm 0{,}044726$ radiants o $\pm 28{,}4735$ graus mil·lessimals).

Així que, si hem decidit de mantenir com a paràmetre on considerar les toleràncies les magnituds angulars, caldrà usar toleràncies trigonomètriques variables, si més no en relació al valor **1**. Per estendre l'anàlisi al rang previsible de toleràncies angulars (**Q0 = 0,1**, **0,01**, **0,001** i **0,0001**), ens ajudarem de la funció

```
(defun SINA (A) (abs (- (sin A) (sin (+ A Q0)))))
```

i compondrem la següent taula de valors:

Q0	(SINA 0)	(SINA (/ PI 2))	(/ (SINA (/ PI 2)) (SINA 0))
0,1	0,0998334	0,00499583	0,05
0,01	0,00999983	0,0000499996	0,005
0,001	0,001	0,0000005	0,0005
0,0001	0,0001	0,000000005	0,00005

Aquesta tolerància, aplicable a les funcions sinus i cosinus quan valen **±1** i que correspon a la tolerància angular **Q0**, pot calcular-se com si féssim **(SINA (PI/2))** (**PI/2** val **(/ PI 2)** i, substituint **(sin PI/2)** pel seu valor **1**, la deixarem reduïda a **(- 1 (sin (+ PI/2 Q0)))**) o bé, si considerem que els valors de l'última columna són **(* 0.5 Q0)**, multiplicant-los pel valor **(SINA 0)** (quedarà **(* 0.5 Q0 (sin Q0))**, valor que si volguéssim podríem aproximar a **(* 0.5 Q0 Q0)**). En el codi que segueix l'hem anomenada **Q90** (les definicions alternatives figuren a les línies precedents, neutralitzades pel ";" inicial) i la utilitzarem exclusivament en la determinació de **NORM-XY** i **NORM-Z**:

```
(defun INSERTOK (2D / K0 Q0 Q90 2*PI PI/2 ... M N W NORM-XY NORM-Z OO-1 OO-2 OO-4)
  (setq K0 9.99994e-11
        Q0 0.001 M "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq M (strcat M "#")))
```

```
(setq 2*PI (* PI 2) PI/2 (/ PI 2)
      V>15 (> (atoi (substr (getvar "ACADVER") 1 2)) 15) ...)
;     Q90 (* 0.5 Q0 Q0)
;     Q90 (* 0.5 Q0 (sin Q0))
      Q90 (- 1 (sin (+ PI/2 Q0))) ...)
(while W
  ..........................................
  (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))
        I (distance '(0 0 0) UV) J (/ I (* OX OY))
        Z (if 2D (mapcar '+ O UV) Z) OZ (if 2D OX OZ)
        K (if 2D 1 (/ (apply '(lambda (X Y Z) (+ X Y Z))
                             (mapcar '* UV (mapcar '- Z O)))
                      (* I OZ)))
        W (if (<= J Q0)
              " e Y están alineados."
              (if (and (not 2D) (equal K 0 Q0))
                  ", Y y Z son coplanarios.")))
  (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
(setq NORM-XY (equal J 1 Q90)
      NORM-Z (or (equal K 1 Q90) (equal K -1 Q90))
      X-Y (expt (distance X Y) 2)
      X-O-Y (+ (expt OX 2) (expt OY 2))
      BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX))) ..............)
```

Resumint el sentit d'aquests canvis, diguem que eren dos els perills d'acoblament defectuós entre l'ús de la tolerància de **0,637** graus mil·lessimals en els controls inicials (independència lineal de **O**, **(X)**, **(Y)** i **(Z)**, i detecció de les situacions **NORM-XY** i **NORM-Z**) i l'adopció de **1** grau mil·lesimal com a intèrval discriminador en els paràmetres α_1, β_1 i β_2 que intervenen en la identificació dels genèrics: d'una banda que la primera fos massa petita, corrent el risc que les situacions singulars esmentades no fossin reconegudes i poguéssin aparèixer monstres com ara *B_397556_342000* o *B_397556_3421000* (sense ser **NORM-XY**), o bé com *B_397000_000333* o *B_3971000_100033* (sense ser **NORM-Z**); de l'altra, que fos massa gran, corrent el risc que certes situacions fossin qualificades de **NORM-XY**, de **NORM-Z** o d'ambdues alhora, amb el doble resultat d'un encaix deficient de *B_397556_342000*, *B_000333* o *B_AMB_ATRIBSTIPS* (dels cubs de referència) en paral·lelepípedes que notòriament no tenen la base quadrada, no són rectes o no són ortogonals, i de sèries incompletes en què el rang de α_1, β_1 o β_2 va de *007* a *993*, per exemple. L'artifici de forçada conversió d'angles de **000** a **001** i de **1000** a **999** graus mil·lessimals és el que ens deixa a recer del primer perill. Quant al segon, l'autor no té el més mínim pudor a subscriure que el métode empíric "és tan bo com d'altres i millor que molts", parafrasejant la lliçó de manual del bon pistoler que Alan Ladd li dona al petit Brandon de Wilde a "Shane" (versionada a l'espanyol com a "Raíces profundas"), sempre i quan hi dediquem prou temps i siguem conscients dels límits dintre dels quals hi ha una raonable seguretat que el sistema no falli: en el cas present, la comprovació s'ha fet amb **Q0 = 0,001** i suposant que entre les longituds **O–(Y)** i **O–(X)**, i entre **O–(Z′)** i **O–(XY′) = O–(X′)**, la relació máxima és de **50** a **1**.

Com a resultat dels canvis de denominació adoptats i de les esmenes introduïdes per harmonitzar (amb les reserves enunciades) els diversos controls dimensionals, la sinopsi que oferíem en el capítol precedent ara tindria l'aspecte següent:

INS3D:

1 Encaix: paral·lelepípede oblic de base romboïdal (ni **NORM-XY** ni **NORM-Z**)

B_SENSE_ATRIBS	*B_SENSE_ATRIBS_397556*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B	*ATRIBS_DE_B_397556*	*ATRIBATIP_2_DE_B*
	B_397556_342333	*ATRIBATIP_3_DE_B*
		ATRIBATIP_1_DE_B_000333
		ATRIBATIP_2_DE_B_000422
		ATRIBATIP_3_DE_B_000029
		ATRIBSATIPS_DE_B_397556_342333

2 Encaix: paral·lelepípede oblic de base rectangular (**NORM-XY**, però no **NORM-Z**)

B_SENSE_ATRIBS	*B_SENSE_ATRIBS_397556*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B	*ATRIBS_DE_B_397556*	*ATRIBATIP_2_DE_B*
	B_397556_342000	*ATRIBATIP_3_DE_B*
		ATRIBATIP_2_DE_B_000438
		ATRIBATIP_3_DE_B_000140
		ATRIBSATIPS_DE_B_397556_342000

3 Encaix: paral·lelepípede recte de base romboïdal (no **NORM-XY**, però sí **NORM-Z**)

B_SENSE_ATRIBS	*B_000333*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B		*ATRIBATIP_2_DE_B*
		ATRIBATIP_3_DE_B
		ATRIBATIP_1_DE_B_000333
		ATRIBATIP_3_DE_B_000820
		ATRIBSATIPS_DE_B_000333

INS2D:

5 Encaix: paral·lelogram oblic (no **NORM-XY**, però sí **NORM-Z**)

B_SENSE_ATRIBS	*B_000333*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B		*ATRIBATIP_2_DE_B*
		ATRIBATIP_3_DE_B
		ATRIBATIP_1_DE_B_000333
		ATRIBATIP_3_DE_B_000798
		ATRIBSATIPS_DE_B_000333

I bé: ¿tanta història només per deixar palesa en alguns casos la continuïtat de la sèrie *B_397556_342000*, *B_397556_342001*, *B_397556_342002* ..., per exemple, en què ni tan sols s'apreciaria continuïtat morfològica? No, certament, perquè ja havíem deixat entreveure objectius més ambiciosos: posar de manifest la redundància que encara s'amaga en el sistema i abordar una notable simplificació en l'estructura arbòria virtual formada pels genèrics teòricament disponibles que pengen de **BLOC**. No variarà el nombre de nivells de l'arbre però sí el nombre de genèrics **BLOC****, que baixarà de $\frac{1}{Q0^4}$ a $\frac{1}{Q0^3}$, amb una modificació paral·lela en la manera d'anomenar-los. Tot i que els criteris que regiran les denominacions futures faran que els paral·lelepípedes d'encaix reflectits en l'exemple es manifestin en nous valors numèrics, en el llistat que oferim ara mantenim els d'abans, amb l'únic propòsit de mostrar amb més claredat en què consisteix la simplificació:

INS3D:

1 Encaix: paral·lelepípede oblic de base romboïdal (ni **NORM-XY** ni **NORM-Z**)

B_SENSE_ATRIBS	*B_SENSE_ATRIBS_397556*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B	*ATRIBS_DE_B_397556*	*ATRIBATIP_2_DE_B*
	B_397556_333	*ATRIBATIP_3_DE_B*
		ATRIBATIP_1_DE_B_333
		ATRIBATIP_2_DE_B_422
		ATRIBATIP_3_DE_B_029
		ATRIBSATIPS_DE_B_397556_333

2 Encaix: paral·lelepípede oblic de base rectangular (**NORM-XY**, però no **NORM-Z**)

B_SENSE_ATRIBS	*B_SENSE_ATRIBS_397556*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B	*ATRIBS_DE_B_397556*	*ATRIBATIP_2_DE_B*
	B_397556_000	*ATRIBATIP_3_DE_B*
		ATRIBATIP_2_DE_B_438
		ATRIBATIP_3_DE_B_140
		ATRIBSATIPS_DE_B_397556_000

3 Encaix: paral·lelepípede recte de base romboïdal (no **NORM-XY**, però sí **NORM-Z**)

B_SENSE_ATRIBS	*B_333*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B		*ATRIBATIP_2_DE_B*
		ATRIBATIP_3_DE_B
		ATRIBATIP_1_DE_B_333
		ATRIBATIP_3_DE_B_820
		ATRIBSATIPS_DE_B_333

INS2D:

5 Encaix: paral·lelogram oblic (no **NORM-XY**, però sí **NORM-Z**)

B_SENSE_ATRIBS	*B_333*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B		*ATRIBATIP_2_DE_B*
		ATRIBATIP_3_DE_B
		ATRIBATIP_1_DE_B_333
		ATRIBATIP_3_DE_B_798
		ATRIBSATIPS_DE_B_333

Com haureu observat, en tots dos llistats ens hem estalviat de reproduir els casos 4 (Encaix: paral·lelepípede ortogonal (**NORM-XY** i **NORM-Z**)) i 6 (Encaix: rectangle (**NORM-XY** i **NORM-Z**)) perquè, en assolir-se l'encaix amb la inserció directa dels blocs portadors d'atributs atípics, de *B_AMB_ATRIBSTIPS* i de *ATRIBSATIPS_DE_B*, no afectava gens ni mica la problemàtica que estàvem tractant.

¿Qué potser es tracta de cruspir-nos la referència a β_2? Doncs sí: és justament el que hem fet i, d'alguna manera, la decisió d'ampliar la denominació *B_397556_342* a *B_397556_342000* obeïa també á la conveniència (no la única ni la principal) de poder-la deixar-la ara en *B_397556_000*, i no amb un aspecte tan inacabat (i tan perillosament semblant a *B_397556*) com *B_397556_*; pel que fa a *B_000333*, atès que els tres zeros responien a $\beta_2 = 0$, el resultat d'aquesta acció serà el retorn a l'aspecte previ a l'últim criteri onomàstic, i es quedarà en *B_333*. Però, abans de passar a la justificació geomètrica d'aquesta decisió tan dràstica, i precisament per no entorpir el discurs amb més digressions, plantegem una última objecció a la historia dels zeros: seria força assenyat admetre la poca contundència formal d'un identificador com *B_397556_*, però argumentar que encara seria més net deixar-ho en *B_397556_333*, *B_397556* i *B_333*, perquè només comparant el nombre de tríades de xifres ja es veu que mai a la vida podrà haver-hi confusió entre els tres noms. L'argument és incontestable i a més aquest sistema onomàstic seria el més econòmic en caràcters, però l'autor té una especial obsessió (potser perquè els dubtes al respecte l'han fet perdre temps) en la semàntica que gairebé sense adonar-nos hem anat configurant a propòsit d'una estructura de noms com **"*"**, **"*_###"**, **"*_######"**, i **"*_######_###"** (on prèviament s'hauria d'especificar que la cadena de caràcters representada amb **"*"** no inclou tríades numèriques): és prou evident que, posats a fer una primera classificació, tothom diria que el perfil **"*"** respon al nom d'un bloc **BLOC** o dels substituts **BLOC-1** i **BLOC-2**, **"*_###"** i **"*_######"** ho fan al nom de genèrics de primer grau **BLOC*** o dels substituts genèrics de primer grau **BLOC-1*** i **BLOC-2***, i que únicament **"*_######_###"** correspon a genèrics de segon grau **BLOC**.** Doncs bé: **"*_######_000"** no és altra cosa que un cas particular de **"*_######_###"**, cosa que pot justificar un tractament singular però que no treu la seva condició de genèric de segon grau (inserció d'una inserció de **BLOC**); escapçar-li el sufix **"_000"** i deixar-lo en **"*_######"** es presta a massa malentesos, el pitjor d'ells prendre'l per un genèric de primer grau (inserció simple de **BLOC**). Aclarit això, anem al gra.

Quan en el capítol ENCAIX 3D DE BLOCS AMB ATRIBUTS vam decidir d'aplicar el mètode de Ritz directament al paral·lelogram (XY')–O–(Z'), en el sistema de referència "B", sense ser-ne conscients estàvem perdent l'oportunitat de simplificar sensiblement el procés. D'altra banda, no feiem altra cosa que seguir la pauta marcada en el sistema "A2": en el "B2" reuniem els paral·lelograms en subconjunts caracteritzats per resultar de transformacions afins en què el quadrat i l'eix (en definitiva, l'angle β_1) eren els mateixos; dins de cada subconjunt, la raó d'afinitat donada per $\dfrac{(XY')-(XY')''}{(XY')'-(XY')''} = \dfrac{\tan\beta_2}{\tan\beta_1} = \dfrac{E_y}{E_x}$ (només ens interessa l'última relació, no pas els factors d'escala absoluts) anava variant, cosa que ens obligava a identificar els genèrics **BLOC**** amb β_2, com **BLOC-1*** i **BLOC-2*** ho havien estat amb β_1. Però teniem una altra possibilitat: en comptes d'aplicar el procediment a cada paral·lelogram (XY')–O–(Z'), aplicar-lo a un de representatiu (d'un subconjunt definit d'una altra manera, no pas de l'esmentat) en què la raó d'afinitat tingui un valor depenent exclusivament de l'angle β_1. Dit així sembla que tot plegat és un joc de paraules, perquè si fem una reducció substituint una sèrie de paral·lelograms per un d'ells, obtenint-ne el quadrat, l'eix i una raó d'afinitat, tard o d'hora caldrà recuperar la diversitat original, introduint al final els elements diferenciadors que havíem ignorat en començar. Per centrar la discusió, si ara considerem paral·lelograms oblics que tenen en comú la relació entre la base i la projecció sobre ella dels costats veïns, ¿què més dóna treballar directament amb les figures de primera mà, obtenint de Ritz la relació $\dfrac{E_y}{E_x}$ de factors d'escala que ens permetrà de tractar-los com a insercions d'un quadrat girat un determinat angle β_1, que substituir-los per per un paral·lelogram que comparteixi la amb els precedents la característica esmentada i que, a més, tingui una determinada relació base/altura? Si allò que pretenem segueix sent poder restituir els paral·lelograms originals mitjançant la inserció d'un bloc quadrat, al primer cop d'ull pot semblar que la feina del segon procés és semblant a la del primer i no en treurem cap profit perquè, a banda de verificar si s'acompleix la relació que identifica el conjunt, caldrà prendre nota de la raó $\dfrac{E'_x}{E'_y}$ entre la relació base/altura de cada paral·lelogram i la d'aquell que havíem triat com a representant: si aquest es pot obtenir del bloc quadrat amb

els factors d'escala $\mathbf{E''_x}$ i $\mathbf{E''_y}$, la relació de factors que permetrà restituir cada un dels demés seria $\frac{\mathbf{E_y}}{\mathbf{E_x}} = \frac{\mathbf{E'_y}}{\mathbf{E'_x}} \times \frac{\mathbf{E''_y}}{\mathbf{E''_x}}$; així que només hi hauria hagut una transposició en l'ordre de les operacions, cosa que complicaria innecessàriament el processat. Però a qui n'hagués tret aquesta opinió se li hauria escapat un petit detall: que la inserció amb l'escalat diferencial $\frac{\mathbf{E'_y}}{\mathbf{E'_x}}$ no es produeix el el mateix sistema "B2" (amb el mateix eix d'afinitat) que la inserció del bloc quadrat amb $\frac{\mathbf{E''_y}}{\mathbf{E''_x}}$ per donar lloc al paral·logram representatiu, raó per la qual la conclusió $\frac{\mathbf{E_y}}{\mathbf{E_x}} = \frac{\mathbf{E'_y}}{\mathbf{E'_x}} \times \frac{\mathbf{E''_y}}{\mathbf{E''_x}}$ és errònia; la primera es produeix en el sistema "B" i aquesta circumstància, junt al fet que l'eix ***Y*** del sistema "B" coincideix amb l'eix ***Z*** dels sistemes "A2" i "A", i al caràcter de resultat final de la inserció de **BLOC**** (ni els factors d'escala ni l'angle de gir no queden registrats enlloc), és allò que ens permetrà ajornar fins a aquest moment l'aplicació de les escales $\mathbf{E'_x}$ i $\mathbf{E'_y}$, amb la ruptura de vincles amb β_2 que això comporta (el nou paral·lelogram ja no dependrà de β_1 i β_2, sinó només de l'angle β''_1). Però ja tornarem, per justificar-la millor, a la conseqüència més transcendent del nou enfoc. De moment, ens interessa remarcar que els subconjunts de paral·lelograms propis del mètode vigent fins ara i els del mètode alternatiu que proposem no coincideixen, ja ho hem dit però cal insistir-hi: cada subconjunt d'ara estaria format per un paral·lelogram de cadascun dels subconjunts antics, precisament aquells en què la relació entre la base i la projecció sobre ella dels costats veïns tingui un valor determinat.

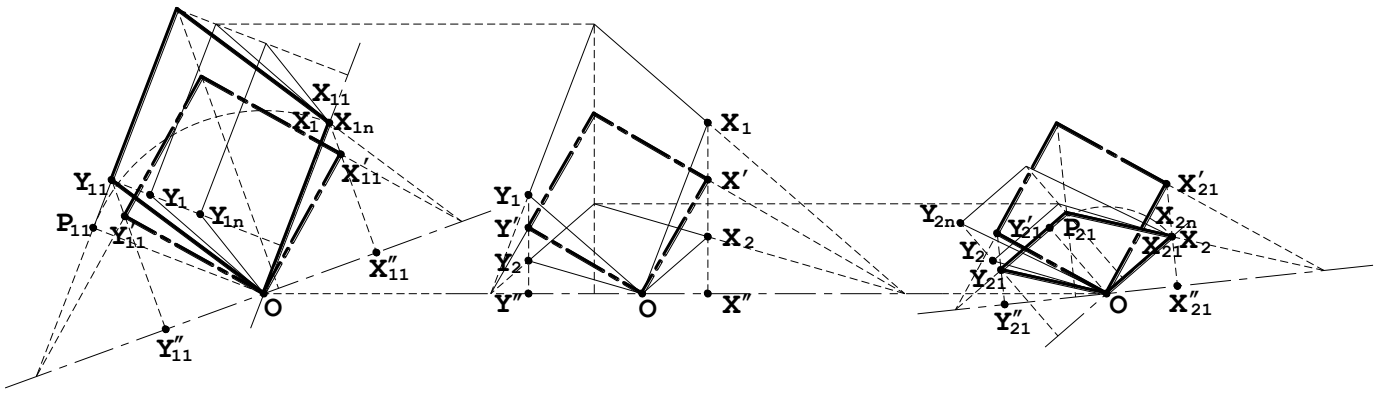


Figura 12.2

Encara permetrà de simplificar més el procés concretar la relació base/altura en la unitat, és a dir, definir com a paral·lelogram representatiu de cada subconjunt aquell en què base = altura. D'ara endavant, a aquest paral·lelogram l'anomenarem "romboide estàndard" (romboide perquè, tret del cas particular del quadrat, mai no podrà ser un rombe). La Figura 12.2 (on, per generalitzar la validesa del discurs i de passada simplificar la notació, ens oblidarem momentàniament que la discussió se centrava en les figures $\mathbf{(XY')-O-(Z')}$ i $\mathbf{(XY')'-O-(Z')'}$) reflecteix tot el que deiem: en el centre tenim una sèrie de paral.lelograms transformats del quadrat $\mathbf{X'-O-Y'}$ respecte a l'eix $\mathbf{X''-Y''}$, representats per $\mathbf{X_1-O-Y_1}$ i $\mathbf{X_2-O-Y_2}$; a l'esquerra, on hem traslladat el paral·lelogram $\mathbf{X_1-O-Y_1}$, hem dibuixat el romboide estàndard $\mathbf{X_{11}-O-Y_{11}}$ ($\mathbf{O-X_{11} = O-P_{11}}$) representatiu d'ell i d'altres paral·lelograms que comparteixen la base $\mathbf{O-X}$ i hi tenen el vèrtex $\mathbf{Y}$ projectat en el mateix punt, com ara $\mathbf{X_{1n}-O-Y_{1n}}$, i el quadrat $\mathbf{X_{11}'-O-Y_{11}'}$ del qual surt el romboide estàndard per transformació afí d'eix $\mathbf{X_{11}''-Y_{11}''}$; a la dreta, on hem traslladat el paral·lelogram $\mathbf{X_2-O-Y_2}$, hi ha dibuixat el romboide estàndard $\mathbf{X_{21}-O-Y_{21}}$ ($\mathbf{O-X_{21} = O-P_{21}}$) representatiu d'ell i d'altres paral·lelograms que comparteixen la base $\mathbf{O-X}$ i hi tenen el vèrtex $\mathbf{Y}$ projectat en el mateix punt, com ara $\mathbf{X_{2n}-O-Y_{2n}}$, i el quadrat $\mathbf{X_{21}'-O-Y_{21}'}$ del qual

surt el romboide estàndard per transformació afí d'eix $\mathbf{X_{21}''–Y_{21}''}$. Observeu que les sèries $\mathbf{X_{11}–O–Y_{11}}$ ... $\mathbf{X_1–O–Y_1}$ ... $\mathbf{X_{1n}–O–Y_{1n}}$ i $\mathbf{X_{2n}–O–Y_{2n}}$ ... $\mathbf{X_2–O–Y_2}$ ... $\mathbf{X_{21}–O–Y_{21}}$ no tenen res a veure amb la parella $\mathbf{X_{11}–O–Y_{11}}$ i $\mathbf{X_{11}'–O–Y_{11}'}$, la parella $\mathbf{X_{21}–O–Y_{21}}$ i $\mathbf{X_{21}'–O–Y_{21}'}$, ni amb la sèrie $\mathbf{X_1–O–Y_1}$ ... $\mathbf{X'–O–Y'}$ ... $\mathbf{X_2–O–Y_2}$, i que l'escalat $\mathbf{E_y}$ que relaciona els elements de les dues primeres sèries actua perpendicularment a les respectives bases, que coincideixen amb l'eix ***X*** del sistema "B", mentre que en l'última sèrie (mètode antic) o en les dues parelles romboide estàndard / quadrat (nou mètode) actua perpendicularment a l'eix ***X*** del sistema "B2".

En passar el genèric **BLOC**** a representar una inserció de **BLOC*** en què el vèrtex $\mathbf{(Z')}$ del prisma oblic de base quadrada s'ha desplaçat verticalment per igualar la seva altura a la longitud $\mathbf{O–(X')}$ del costat d'aquest quadrat, en general variarà la inclinació i la longitud de l'aresta $\mathbf{O–(Z')}$, així que els angles $\boldsymbol{\beta_1 = (XY')''–O–(XY')'}$ i $\boldsymbol{\beta_2 = (XY')–O–(XY')''}$ també hauran variat. Per exemple, el bloc *B* del capítol precedent ara necessitaria els genèrics *B_397586* i *B_397586_810866*, en comptes de *B_397643* i *B_397643_859866*, en les condicions **(not (or NORM-XY NORM-Z))** (queda clar que en aquestes condicions d'encaix no usem **BLOC*** i que en realitat volem representar *B_SENSE_ATRIBS_397586* i *ATRIBS_DE_B_397586*), i de nou *B_397586* i *B_397586_810*, en comptes de *B_397643* i *B_397643_859*, en les condicions **(and NORM-XY (not NORM-Z))**. Només quan tinguéssim **(and (not NORM-XY) NORM-Z)** el genèric *B_866* seria el mateix.

Però és que la reducció de tots els prismes oblics que tenen una mateixa projecció de $\mathbf{(Z')}$ sobre el pla de la base quadrada $\mathbf{(X')–O–(Y')}$ al seu prisma estàndard (si ens és permès d'extrapolar-hi la denominació donada inicialment als paral·lelograms d'alçada $\mathbf{O–(X')}$) no era cap caprici orientat només a reorganitzar l'adscripció dels genèrics **BLOC*** a diversos casos d'encaix sobre la base d'un criteri més fàcilment identicable per l'usuari (és una simple reorganització en què, si bé d'una banda tenim que els antics *B_397399*, *B_397643* i *B_397796* s'agrupen en un únic *B_397586*, de l'altra els casos que abans compartien un mateix *B_397643* ara es diversifiquen, com *B_397586*, *B_397766*, etc.), sinó en substituir efectivament cada subconjunt de genèrics **BLOC**** derivats d'un mateix **BLOC*** (potencialment, de **(/ 1 Q0)** elements) per un únic **BLOC****: el que correspon al seu prisma estàndard. Que això és possible no només s'intueix des de casos concrets (en el primer dels exemples precedents, els antics *B_397399_633866*, *B_397643_859866* i *B_397796_975866* convergeixen sobre *B_397586_810866*, cosa que ens fa pensar si no n'hi hauria prou amb *B_397586_866*), sinó que és conclusió obligada si pensem que el penúltim paràmetre implícit en el nom de **BLOC**** és l'angle $\boldsymbol{\beta_2}$ que ara queda determinat per $\boldsymbol{\beta_1}$ (és a dir, per **BLOC***). Només cal pensar en el mètode de Ritz estricte (en el sentit que donàvem al mot en el capítol NOUS RECURSOS: ENCAIX 3D DE BLOCS SENSE ATRIBUTS) per deduir que a cada romboide estàndard li correspon un quadrat i un eix definidors de la transformació d'afinitat que relaciona la segona figura amb la primera. Tanmateix ens assalta un dubte: donat un quadrat i un eix, ¿pot ser que que hi hagi més d'una transformació afí que doni com a resultat un romboide estàndard? De seguida veurem que no, però abans volem fer una precisió relativa al nomenclàtor que farem servir: tot i que les transformacions considerades se situen en el sistema de referència "B" i que, si mantinguéssim les denominacions usades en els capítols precedents, hauríem de parlar del quadrat $\mathbf{XY'–O–Z'}$ (o $\mathbf{(XY')'–O–(Z')'}$, per ser fidels a aquella notació) i del paral·lelogram $\mathbf{XY–O–Z}$ (o $\mathbf{(XY')–O–(Z')}$), per no fer-ho més complicat els anomenarem $\mathbf{X'–O–Y'}$ i $\mathbf{X–O–Y}$ respectivament, com ja hem fet més d'una vegada; per les mateixes raons, l'ús dels noms $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ i $\boldsymbol{\gamma}$ per als angles no té res a veure amb els $\boldsymbol{\alpha}$, $\boldsymbol{\alpha_1}$, $\boldsymbol{\alpha_2}$, $\boldsymbol{\beta}$, $\boldsymbol{\beta_1}$, $\boldsymbol{\beta_2}$, $\boldsymbol{\gamma}$, $\boldsymbol{\gamma_1}$ i $\boldsymbol{\gamma_2}$ que havíem vingut utilitzant. Veiem ara com per a cada quadrat i eix hi ha un únic romboide estàndard.

Si partim d'un quadrat en què $\mathbf{O–X' = O–Y' = 1}$ i anomenem $\boldsymbol{\alpha}$ i $\boldsymbol{\beta}$ els angles que formen $\mathbf{O–X'}$ i $\mathbf{O–X}$, respectivament, amb l'eix $\mathbf{O–X''}$, serà $\cos\boldsymbol{\beta} = \dfrac{\mathbf{O–X''}}{\mathbf{O–X}} = \dfrac{\cos\boldsymbol{\alpha}}{\mathbf{O–X}}$.

Així doncs, $\mathbf{O–X} = \dfrac{\cos\boldsymbol{\alpha}}{\cos\boldsymbol{\beta}}$. A la Figura 12.3 hem prolongat fins a tallar l'eix els costats del quadrat aliens al punt **O**, obtenint les interseccions **U** i **V** (són els

punts dobles de les rectes $U-X'$ i $V-Y'$ en totes les tranformacions d'eix $X''-Y''$).

En el triangle rectangle $O-Y'-V$, $\frac{O-Y''}{Y'-Y''} = \frac{Y'-Y''}{Y''-V'}$, és a dir $Y''-V' = \frac{(Y'-Y'')^2}{O-Y''} = \frac{\cos^2\alpha}{\sin\alpha}$

D'altra banda, $O-V = O-Y'' + Y''-V = \sin\alpha + \frac{\cos^2\alpha}{\sin\alpha} = \frac{\sin^2\alpha + \cos^2\alpha}{\sin\alpha} = \frac{1}{\sin\alpha}$

Situant a O l'origen de coordenades, les equacions de les rectes $V-Y$ i $O-Q$ són:

$$y = x\tan\beta + O-V\tan\beta = x\tan\beta + \frac{\tan\beta}{\sin\alpha}$$

$$y = -\frac{x}{\tan\beta}$$

Aplegant-les en un sistema, les coordenades de la seva intersecció Q seran:

$$x = -y\tan\beta$$

$$y = -y\tan^2\beta + \frac{\tan\beta}{\sin\alpha} \rightarrow y\left(1+\tan^2\beta\right) = \frac{\tan\beta}{\sin\alpha} \rightarrow \frac{y}{\cos^2\beta} = \frac{\tan\beta}{\sin\alpha}$$

És a dir que:

$$y = \frac{\cos^2\beta \quad \tan\beta}{\sin\alpha} = \frac{\sin\beta \quad \cos\beta}{\sin\alpha}$$

$$x = -y\tan\beta = -\tan\beta\,\frac{\sin\beta\,\cos\beta}{\sin\alpha} = -\frac{\sin^2\beta}{\sin\alpha}$$

La distància $O-Q$ serà:

$$\frac{\sqrt{\sin^4\beta + \sin^2\beta\,\cos^2\beta}}{\sin\alpha} = \frac{\sin\beta}{\sin\alpha}\sqrt{\sin^2\beta+\cos^2\beta} = \frac{\sin\beta}{\sin\alpha}$$

Així doncs, la relació $\frac{O-X}{O-Q}$ serà $\frac{\frac{\cos\alpha}{\cos\beta}}{\frac{\sin\beta}{\sin\alpha}} = \frac{\sin\alpha\,\cos\alpha}{\sin\beta\,\cos\beta}$

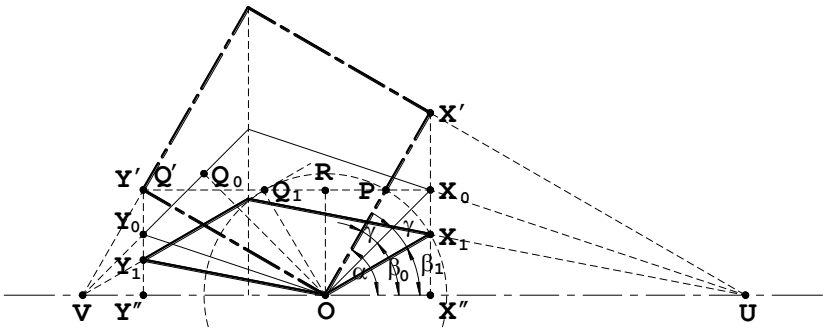


Figura 12.3

Per a valors constants $0° < \alpha < 90°$, representant en coordenades cartesianes la funció $f(\beta) = \frac{\sin\alpha\,\cos\alpha}{\sin\beta\,\cos\beta}$, considerada en l'intèrval $0° \le \beta \le 90°$, és simètrica respecte a l'eix vertical $\beta = 45°$, tendeix a $+\infty$ en els extrems, presenta un mínim $\frac{\sin\alpha\,\cos\alpha}{0{,}5} = 2\sin\alpha\,\cos\alpha$ en $\beta = 45°$ i talla la recta horitzontal $\frac{\sin\alpha\,\cos\alpha}{\sin\beta\,\cos\beta} = 1$ en dos punts: $\beta = \alpha$ (la solució trivial: el quadrat $X'-O-Y'$) i $\beta = 90° - \alpha$ (la que cercàvem: el romboide estàndard X_1-O-Y_1). En el cas que $\alpha = 45°$ només hi haurà una solució (el quadrat $X'-O-Y'$), corresponent al mínim $\beta = 45°$, perquè l'horitzontal $\frac{2}{\sin\beta\,\cos\beta} = 1$ és tangent a $f(\beta)$ en aquest punt.

A la Figura 12.3 hem representat precisament el romboide corresponent a $\beta_0 = 45°$ on $\frac{O-X}{O-Y}$ és mínim (X_0-O-Q_0), l'estàndard $\beta_1 = 90°-\alpha$ on $\frac{O-X}{O-Y} = 1$ (X_1-O-Q_1) i prou, per no complicar-la: no hem dibuixat cap altre paral.lelogram afí amb el quadrat i

l'eix, de manera que, quan més endavant ens referim als punts **X** i **Y** amb caràcter genèric, el lector haurà de seguir l'explicació sobre X_0 i Y_0, o sobre X_1 i Y_1. Gràficament és immediat de construir-los a partir del quadrat **X′–O–Q′** i de l'eix d'afinitat **X″–O–Q″**: n'hi ha prou a dibuixar una recta paral·lela a **X″–O–Q″** des del vèrtex veí de **O** més proper a aquest eix (**Y′**, a la figura); el punt **P** on talli el quadrat determinarà la longitud **O–P** de la base del romboide estàndard, que caldrà traslladar a $O–X_1$; el punt X_0 on talli la recta **X′–X″** ja ens donarà la base del paral·lelogram de relació base/altura mínima (o altura/base màxima, com vulgueu).

Com que ja ho hem apuntat analíticament (de fet no hem arribat a derivar la funció $f(\beta)$, però el lector escèptic sempre ho pot verificar), no ens complicarem la vida demostrant que el paral·lelogram de minima relació base/altura queda determinat per la intersecció X_0 entre **X′–X″** i una recta que forma **45°** amb l'eix, sinó que ens limitarem a veure que la intersecció entre la primera i la paral·lela a l'eix des de **Y′** hi coincideix: els triangles rectangles **O–X″–X′** i **Y′–Y″–O** són congruents per ser **Y′–Y″** perpendicular a **O–X″**, **Y″–O** a **X″–X′**, **O–Y′** a **X′–O** i ser aquests dos últims iguals; i com que $X_0–X'' = Y'–Y'' = O–X''$, $\beta_0 = 45°$.

Demostrar que la intersecció **P** del costat **O–X′** amb la paral·lela a l'eix des de **Y′** determina un paral.lelogram que és el romboide estàndard, i que l'angle β_1 és el complementari de α, ja no ho és tant:

- Si, amb centre a **O** dibuixem una circumferència que passi per aquest punt i talli la recta **X′–X″** a X_1, serà $O–X_1 = O–P$. L'altre punt on aquesta circumferència talla la recta $X_0–Y'$ serà Q_1, simètric de **P** respecte a la perpendicular a $X_0–Y'$ per **O** (mediatriu de $P–Q_1$), cosa que comporta $O–Q_1 = O–P = O–X_1$.
- Per ser **Y′–O** perpendicular a **O–X′** i **Y′–Q** a **X′–X₁**, els angles $O–Y'–X_0$ i **O–X′–X** són iguals.
- Els triangles $O–X_1–X'$ i $O–Q_1–Y'$ són congruents, per ser $O–Y' = O–X'$, $O–Q_1 = O–X_1$ i tenir igual l'angle oposat a aquest últim*.
- Com que són congruents, i **O–Y′** és perpendicular a **O–X′** i **Y′–Q₁** ho és a **X′–X₁**, també seran perpendiculars $O–Q_1$ i $O–X_1$. Això, juntament amb $O–Q_1 = O–X_1$, vol dir que $X_1–O–Y_1$ és un romboide estàndard.
- Si **R** és la intersecció amb $P–Q_1$ de la mediatriu d'aquesta corda, els triangles rectangles **O–X″–X′** i **O–R–Y′** seran congruents. Com que $R–P = R–Q_1$, per simetria respecte a **O–R**, serà $R–P = X''–X_1$ i, en ser $R–X_0 = O–X'' = X''–X_0$, $X_0 – P = X_0–X''$.
- Els triangles $O–R–X_0$ i $O–X_1–X_0$ seran simètrics respecte a $O–X_0$, per compartir aquest costat i tenir iguals els altres dos. Si anomenem γ l'angle $X_1–O–X_0 = X_0–O–X'$, l'angle **X″–O–X′** és $\alpha = 45°– \gamma$ i l'angle $X''–O–X_1$ és $\beta_1 = 45°+ \gamma$, d'on sumant ambdues igualtats deduïm que $\alpha + \beta_1 = 90°$ i $\beta_1 = 90°– \alpha$.

Quedi tota aquesta digressió com a simple exercici de geometria recreativa, perquè el problema no és trobar el romboide estàndard, en el sistema de referència "B", a partir del paralel·logram primari **(XY′)–O–(Z′)**, sinó determinar el quadrat i l'eix d'afinitat a partir d'un dels dos. I, en aquest sentit, no hi ha cap procediment abreujat que permeti evitar **CALCULA** (transcripció algorísmica de la construcció de Ritz), funció que de tota manera ja teníem i haurem d'utilitzar en el sistema "A". No surt gens a compte anar a **CALCULA** des del paralel·logram primari per trobar el quadrat i l'eix, i després obtenir el romboide estàndard aplicant d'alguna manera

---

* En realitat, els tres casos "rodons" de congruència de triangles són: 1) igualtat de dos costats i de l'angle que formen; 2) igualtat d'un costat i dels dos angles contigus; 3) igualtat dels tres costats. El quart cas ja té una severa restricció: dos triangles són congruents si tenen iguals dos costats desiguals (ela angles oposats, doncs, també són desiguals) i l'angle oposat al més gran d'aquests (que és l'angle més gran dels dos esmentats). ¿Què passa quan l'angle igual és l'oposat al més petit dels costats iguals (el més petit dels angles esmentats)? Per centrar-nos en aquest diguem-ne "cinquè cas", simplifiquem la notació i imaginem triangles **A–B–C** en què els dos costats iguals són **a < b** i l'angle és **A**. Si anomenem **d** la distància del vèrtex **C** a la recta **c** dibuixada des del vèrtex **A** per aplicació de l'angle d'igual nom, i dibuixem una circumferència de radi **a** amb centre a **C**, poden donar-se tres situacions:

- Si **a < d** no podrem triangular: la circumferència no tallarà **c**.
- Si **a = d** la triangulació serà única: la circumferència serà tangent a **c** en **B**.
- Si **a > d** hi haurà dues triangulacions possibles: les dues interseccions B_1 i B_2 se situaran a la mateixa banda del vèrtex **A**, perquè **d < a < b**. Com que $B_1–B_2–C$ és un triangle isòsceles, si anomenem B_1 al vèrtex més allunyat de **A** i B_2 al més proper, l'angle B_1 serà agut i B_2 serà obtús.

La congruència de $O–X_1–X'$ i $O–Q_1–Y'$ s'inscriu en l'últim dels supòsits de l'anomenat "cinquè cas": $O–Q_1$ seria **a**, **O–Y′** seria **b**, **O–R** seria **d** (**d < a < b**, per construcció) i l'angle $O–Y'–Q_1$; dels dos angles B_1 ($Y'–P–O$, agut) i B_2 ($Y'–Q_1–O$, obtús), ens quedem amb el segon.

la construcció gràfica descrita. És millor introduir a **INSERT*** uns retocs mínims per determinar aquest romboide i anar des d'aquí a **CALCULA**, perquè ja només caldrà intervenir en l'assignació múltiple que en precedeix l'accés (quan no és **NORM-Z**), amb les expressions subratllades:

```
(setq X (inters X* Y* O Z-XY* ())
      X (if X X Z-XY*)
      Z-XY (- (angle O X) (angle Y* X*))
      Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
      Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
      OZ (/ (last Z) OX*)
      Z (trans (list (car Z-XY*) (cadr Z-XY*) OX*) 1 0))
```

Tot el procés seguirà igual (tret de la supressió de la referència a β_2 en el nom de **BLOC****, que ja detallarem), i en la inserció final reintroduirem en l'escala E_z el factor **OZ**, per restituir la tercera coordenada:

```
(eval (append '(command "INSERT" BLOC* O "XYZ" OX*
                        (setq J (/ (* (distance Y Y**) OX*) OX**))
                        (setq K (* (if (and NORM-Z (not 2D))
                                       OZ
                                       (* (if 2D 1 OZ) OX*)) I)) X**)
              (if ATREQ WWAA-1)))
```

Tenint en compte que el primer dels factors de **K** valdrà **OZ** quan sigui **(not 2D)**, tant si és **NORM-Z** com si no, aquesta expressió es pot simplificar i finalment la deixarem en:

```
(eval (append '(command "INSERT" BLOC* O "XYZ" OX*
                        (setq J (/ (* (distance Y Y**) OX*) OX**))
                        (setq K (* (if 2D OX* (* OZ (if NORM-Z 1 OX*))) I)) X**)
              (if ATREQ WWAA-1)))
```

Però serà millor reproduir íntegrament les funcions noves i totes les que s'hagin vist afectades en relació a VERSIÓ 22++ (en aquestes hi seguirem subratllant les expressions que varien), inserint entre funció i funció els comentaris que calgui:

```
; fragments de VERSIÓ 23++
```

Les modificacions que havíem proposat en **REDEF-BLOC*S**, que tenían a veure amb el perfil del nom dels genèrics, no s'han de prendre en consideració, perquè el codi de la VERSIÓ 22++ manté plena vigència.

Sí que haurem de tenir en compte l'ampliació de **CALCULA** i la nova funció **ANG**:

```
(defun CALCULA (U V OV AGUT U* U** V** OU* OU** OV**)
 (setq A (polar O (- (angle O V) PI/2) OV)
       W (if AGUT (angle A U) (angle U A))
       B (mapcar '/ (mapcar '+ A U) '(2 2 2))
       W (angle O (polar B W (distance O B))))
 (set U* (inters U (polar U W 1) O B ()))
 (set U** (inters O (polar O (+ W PI/2) 1) U (eval U*) ()))
 (set V** (inters V (polar V W 1) O (eval U**) ()))
 (set OU* (distance O (eval U*)))
 (set OU** (distance O (eval U**)))
 (set OV** (distance O (eval V**))))

(defun ANG (S C NORM / A)
 (setq A (/ (atan S C) PI/2 Q0)
       A (if (< (- (/ 1 Q0) A) 1)
             (1- (/ 1 Q0))
             (if (< A 1) (if NORM 0 1) A)))
 (FIX+ A))
```

La més estricta apreciació d'ortogonalitat, mitjançant la constant **Q90**, no només afecta **INSERTOK** sinó **INSERT****, on la condició **NORM-XY** ya no estarà representada per l'expressió

```
(equal (setq X-Y (expt (distance X Y) 2))
       (setq X-O-Y (+ (expt OX 2) (expt OY 2))) Q0)
```

sinó per **(equal (/ (distance O Z) (* OX OY)) 1 Q90)**. I, ara que els valors **X-Y** i **X-O-Y** ja no es calculen de passada per les dues funcions esmentades, prescindirem d'aquestes variables i el càlcul dels quadrats el passarem directament a **INSERT***, en el quart argument del primer accés a **CALCULA**. Subratllem els canvis a **INSERT****:

```
(defun INSERT** (BLOC X Y Z G / JJ KK SSAA)
  (command "INSERT" (strcat "ATRIBSATIPS_DE_" BLOC) O "XYZ" X Y Z G
           "DESCOMP" (entlast)
           "SCP" "")
  (setq SSAA (ssget "P") JJ 0 KK -1)
  (repeat OO-4
          (setq JJ (1+ JJ) BLOC (strcat "ATRIBATIP_" (itoa JJ) "_DE_" BL)
                LE (entget (cdr (assoc -2 (tblsearch "BLOCK" BLOC))))
                ICVP (cdr (assoc 70 LE))
                WWAA-1 (if (= (logand ICVP 8) 0) (list (car WWAA-2)))
                WWAA-2 (if WWAA-1 (cdr WWAA-2) WWAA-2)
                KK (1+ KK) O (cdr (assoc 10 (entget (ssname SSAA KK))))
                KK (1+ KK) X (cdr (assoc 10 (entget (ssname SSAA KK))))
                KK (1+ KK) Y (cdr (assoc 10 (entget (ssname SSAA KK)))))
          (command "SCP" "3" O X Y)
          (setq OX (distance O X)
                OY (distance O Y) OZ OX
                O '(0 0 0) X (list OX 0 0)
                Y (trans Y 0 1) Z (U*V X Y)
                Z-XY O I (if (< (last Z) 0) -1 1)
                BLOC-1 "BLOC_NUL" BLOC-2 BLOC
                NORM-XY (equal (/ (distance O Z) (* OX OY)) 1 Q90)
                NORM-Z T OO-2 T OO-4 ())
          (if NORM-XY
              (eval (append '(command "INSERT" BLOC O OX OY 0) (if ATREQ WWAA-1)))
              (INSERT*))
          (command "SCP" "PR"))
  (command "SCP" "PR"
           "BORRA" SSAA ""))
```

Respecte a **INSERT***, sols calia afegir que, després d'haver-nos molestat a canviar el fragment

```
    (if NORM-Z
        (setq BLOC* (strcat BLOC "_"))
        (progn
          ....................................................
          (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                              (FIX+ (/ OXY** OXY* Q0))) ...)
          ....................................................
          (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*
                BLOC* (strcat BLOC* "_" (FIX+ (/ OXY** OX* Q0))))))
    (if (not NORM-XY) (setq BLOC* (strcat BLOC* (FIX+ (/ OX** OX* Q0)))))
```

per

```
    (if NORM-Z
        (setq BLOC* BLOC OXY** OX*)
        (progn
          ....................................................
          (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                              (FIX+ (/ (- 1 (/ OXY** OXY*)) Q0))) ...)
          ....................................................
          (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*)))
    (setq BLOC* (strcat BLOC* "_" (FIX+ (/ (- 1 (/ OXY** OX*)) Q0))
                                  (FIX+ (/ (- 1 (/ OX** OX*)) Q0))))
```

la supressió de la referència a β_2 en el nom de **BLOC**** ens obligarà a eliminar les expressions subratllades en l'últim fragment. Així doncs, la funció quedarà així:

```
(defun INSERT* (/ X* X** Y* Y** Z** OX* OX** Z-XY* XY XY* XY** OXY* OXY** ABLOC*
                  OY** OZ**)
  (command "CLAYER" "0"
           "CECOLOR" "PORBLOQUE"
           "CELTYPE" "PORBLOQUE"
           "CELWEIGHT" -2)
  (if NORM-XY
      (setq X* X X** X
            Y* (polar O PI/2 OX)
            Y** O OX* OX OX** OX OY** 0
            Z-XY* (list (car Z) (* (cadr Z) (/ OX OY)) 0))
```

```
    (progn
      (CALCULA X Y OY (< (expt (distance X Y) 2) (+ (expt OX 2) (expt OY 2)))
               'X* 'X** 'Y** 'OX* 'OX** 'OY**)
      (setq Y* (polar O (+ (angle O X*) PI/2) OX*)
            A (inters Z-XY (polar Z-XY W 1) O X** ())
            Z-XY* (polar A (angle A Z-XY)
                         (/ (* (distance A Z-XY) (distance X* X**))
                            (distance X X**))))))
  (if NORM-Z
      (setq BLOC* BLOC)
      (progn
        (setq X (inters X* Y* O Z-XY* ()) X (if X X Z-XY*)
              Z-XY (- (angle O X) (angle Y* X*))
              Z-XY (if (< Z-XY 0) (+ 2*PI Z-XY) Z-XY)
              Z-XY (if (or (equal Z-XY 0 Q0) (equal Z-XY 2*PI Q0)) 0 Z-XY)
              OZ (/ (last Z) OX*)
              Z (trans (list (car Z-XY*) (cadr Z-XY*) OX*) 1 0))
        (command "SCP" "Z" O X
                 "SCP" "X" (* I 90))
        (setq XY (list OX* 0 0) Z (trans Z 0 1))
        (CALCULA XY Z (distance O Z) (> (car Z) 0)
                 'XY* 'XY** 'Z** 'OXY* 'OXY** 'OZ**)
        (setq K (strcat "_" (FIX+ (* (- 1 Q0) (/ Z-XY PI Q0)))
                            (ANG OZ** OXY** ())))
              BLOC* (strcat BLOC K)
              BLOC-1* (strcat BLOC-1 K) BLOC-2* (strcat BLOC-2 K)
              BL*EX (and (tblsearch "BLOCK" BLOC-1*) (tblsearch "BLOCK" BLOC-2*)))
        (command "SCP" "PR" "SCP" "PR")
        (if (not BL*EX)
            (progn
              (INSPARCIAL 1 1 1 X*)
              (command "SCP" "Z" O X
                       "SCP" "X" 90
                       "GIRA" SS "" O XY*
                       "SCP" "Z" O XY**)
              (PRO-DESHACER-I/F)
              (command "SCP" "PR")))
        (setq B BL*EX BLOC-1 BLOC-1* BLOC-2 BLOC-2*)))
  (setq BLOC* (strcat BLOC* " " (ANG OY** OX** NORM-XY)))
  (if OO-4 (setq ABLOC* (strcat "ATRIBSATIPS_DE_" BLOC*)))
  (if (or NORM-Z BL*EX) (setq BL*EX (and (tblsearch "BLOCK" BLOC*)
  (if OO-4 (tblsearch "BLOCK" ABLOC*) T))))
  (if (not BL*EX)
      (progn
        (if NORM-Z
            (INSPARCIAL 1 1 1 X*)
            (progn
              (if B (command "SCP" "Z" O X "SCP" "X" 90))
              (INSPARCIAL (setq J (/ OXY* OX*))
                          (/ (* (distance Z Z**) J) OXY**) 1 XY**)
              (command "SCP" "PR" "SCP" "PR")))
        (command "SCP" "Z" O X**
                 "BLOQUE" BLOC* O SS "")
        (if OO-4 (command "BLOQUE" ABLOC* O SA ""))
        (command "SCP" "PR")))
  (command "CELWEIGHT" GLIN
           "CELTYPE" TLIN
           "CECOLOR" COL
           "CLAYER" CAPA)
  (eval (append '(command "INSERT" BLOC* O "XYZ" OX*
                          (setq J (/ (* (distance Y Y**) OX*) OX**))
                          (setq K (* (if 2D OX* (* OZ (if NORM-Z 1 OX*))) I)) X**)
                (if ATREQ WWAA-1)))
  (if OO-4 (INSERT** BLOC* OX* J K X**)))
```

Pel que fa a **INSERTOK**, ens limitem a presentar sencer el codi que a cavall de les argumentacions havíem vist de forma fragmentària:

```
(defun INSERTOK (2D / K0 Q0 Q90 2*PI PI/2 V>15 CAPA COL TLIN GLIN ECO CTRL--ECO
                      OSN ATDIA ATREQ EXPERT A B UV BL BLOC BLOC* BLOC-1 BLOC-2
                      BLOC-1* BLOC-2* NLOC-1 BL*EX ICVP WWAA WWAA-1 WWAA-2 WW SA
                      SS O X Y Z Z-XY OX OY OZ E LE E* I J K M N W NORM-XY NORM-Z
                      OO-1 OO-2 OO-4)
  (setq K0 9.99994e-11 Q0 0.001 M "")
  (repeat (strlen (itoa (fix (/ 0.1 Q0)))) (setq M (strcat M "#")))
  (setq 2*PI (* PI 2) PI/2 (/ PI 2)
        V>15 (> (atoi (substr (getvar "ACADVER") 1 2)) 15)
  ;     Q90 (* 0.5 Q0 Q0)
  ;     Q90 (* 0.5 Q0 (sin Q0))
        Q90 (- 1 (sin (+ PI/2 Q0)))
        CAPA (getvar "CLAYER")
        COL (getvar "CECOLOR")
        TLIN (getvar "CELTYPE")
        GLIN (getvar "CELWEIGHT")
        ECO (getvar "CMDECHO")
        OSN (getvar "OSMODE")
        ATDIA (getvar "ATTDIA")
        ATREQ (= (getvar "ATTREQ") 1)
        EXPERT (getvar "EXPERT")
        O (getvar "INSNAME")
        BLOC (getstring (strcat "\nIndique nombre de bloque" (DEFECTE O) ": ") T)
        BLOC (if (= BLOC "")
                 (if (> O "") O (prompt "\nNombre de bloque no válido."))
                 (if (or (and (setq O (tblsearch "BLOCK" BLOC))
                              (< (cdr (assoc 70 O)) 4))
                         (and (not O) (findfile (strcat BLOC ".dwg"))))
                     BLOC (alert (if O (REFX) (RUTES)))))
        BLOC (if BLOC (strcase BLOC) (exit)) W T)
  (while W
    (initget 1) (setq O (getpoint "\nPrecise punto de inserción: "))
    (foreach P (if 2D '("X" "Y") '("X" "Y" "Z"))
      (initget 1)
      (setq A (getpoint O (strcat "\nSitue el extremo de un segmento +"  P
                                  " desde el punto de inserción: "))
            W (getdist O (strcat "\n   Longitud de este segmento en el bloque "
                                 BLOC  " <1>: "))
            W (if W W 1))
      (set (read P) (mapcar '(lambda (CO CA) (+ CO (/ (- CA CO) W))) O A))
      (set (read (strcat "O" P)) (/ (distance O A) W)))
    (setq UV (U*V (mapcar '- X O) (mapcar '- Y O))
          I (distance '(0 0 0) UV) J (/ I (* OX OY))
          Z (if 2D (mapcar '+ O UV) Z) OZ (if 2D OX OZ)
          K (if 2D 1 (/ (apply '(lambda (X Y Z) (+ X Y Z))
                               (mapcar '* UV (mapcar '- Z O))) (* I OZ)))
          W (if (<= J Q0)
                " e Y están alineados."
                (if (and (not 2D) (equal K 0 Q0)) ", Y y Z son coplanarios.")))
    (if W (alert (strcat "\nREPITE:\nEl punto de Inserción,\nX" W))))
  (setq NORM-XY (equal J 1 Q90) NORM-Z (or (equal K 1 Q90) (equal K -1 Q90))
        BL BLOC BLOC (if (tblsearch "BLOCK" BLOC) BLOC (DIBUIX)))
  (SEGR-ATRIBS)
  (if (and NORM-XY NORM-Z)
      (progn
        (eval (append '(command "INSERT" NLOC-1 O "XYZ" OX OY (* OZ I) 0)
                      (if ATREQ WWAA-1)))
        (if OO-4 (INSERT** BLOC OX OY (* OZ I) 0)))
      (INSERT*))
  (command "SCP" "PR"
           "INSNAME" BLOC
           "EXPERT" EXPERT
           "ATTDIA" ATDIA
           "OSMODE" OSN
           "DESHACER" "F")
  (setvar "CMDECHO" ECO)
  (princ))
```

Igual que amb **REDEF-BLOC*S**, les modificacions que havíem proposat en **C:DESCOMPOK** (sobre VERSIÓ 22A++ i VERSIÓ 22B++) no s'han de prendre en consideració, perquè el codi de la VERSIÓ 22++ manté plena vigència. I amb aquesta precisió queda descrit el codi de la VERSIÓ 23++. Com heu vist, la majoria dels canvis tenen a veure amb la decisió de substituir en el nom dels genèrics la referència a la funció cosinus per la referència directa a l'angle, i l'ajust de les aproximacions als estats **NORM-XY** i **NORM-Z**. Però la substitució estratègica del paral.lelogram **(XY′)–O–(Z′)** pel seu romboide unitari es concentra en uns canvis puntuals en la funció **INSERT***. Si abans no haguéssim donat el pas d'homologar tots els blocs genèrics a mides unitàries (VERSIÓ 22++), diríem que la simplificació s'hauria materialitzat en el pas d'un genèric **BLOC**** en què el prisma oblic de referència, de base quadrada, era geomètricament semblant a l'original (no parlem del paral·lelepípede d'encaix sinó del previ a l'última inserció), a un altre en què hauria variat la inclinació de l'aresta **O–(Z′)** sobre la base per tal d'igualar l'alçada i el costat de la base: la secció d'aquest prisma per un pla contenidor de l'aresta esmentada i normal a la base ens donaria el romboide estàndard (tret d'una amplada **O–(XY′) = O–(X′)** que no sortia del tall sinó d'un conveni que igualava la base al costat del quadrat). Però, en haver-lo donat, ara podem parlar d'un prisma oblic de base **1 × 1** i alçada **1**, que no és semblant a l'original però en què la projecció del vèrtex **(Z′)** sobre aquesta base sí que ho serà (volem dir que les distàncies des d'aquesta projecció a punts homòlegs en la base mantindran la raó de semblança).

Aclarit això, aplicarem la nova estratègia als casos de referència per actualitzar noms (no només quant a sintaxi, com feiem provisionalment l'últim cop) i tindrem:

INS3D:

1 Encaix: paral·lelepípede oblic de base romboïdal (ni **NORM-XY** ni **NORM-Z**)

B_SENSE_ATRIBS	*B_SENSE_ATRIBS_397601*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B	*ATRIBS_DE_B_397601*	*ATRIBATIP_2_DE_B*
	B_397601_333	*ATRIBATIP_3_DE_B*
		ATRIBATIP_1_DE_B_333
		ATRIBATIP_2_DE_B_422
		ATRIBATIP_3_DE_B_029
		ATRIBSATIPS_DE_B_397601_333

2 Encaix: paral·lelepípede oblic de base rectangular (**NORM-XY**, però no **NORM-Z**)

B_SENSE_ATRIBS	*B_SENSE_ATRIBS_397601*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B	*ATRIBS_DE_B_397601*	*ATRIBATIP_2_DE_B*
	B_397601_000	*ATRIBATIP_3_DE_B*
		ATRIBATIP_2_DE_B_438
		ATRIBATIP_3_DE_B_140
		ATRIBSATIPS_DE_B_397601_000

3 Encaix: paral·lelepípede recte de base romboïdal (no **NORM-XY**, però sí **NORM-Z**)

B_SENSE_ATRIBS	*B_333*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B		*ATRIBATIP_2_DE_B*
		ATRIBATIP_3_DE_B
		ATRIBATIP_1_DE_B_333
		ATRIBATIP_3_DE_B_820
		ATRIBSATIPS_DE_B_333

4 Encaix: paral·lelepípede ortogonal (**NORM-XY** i **NORM-Z**)

		ATRIBATIP_1_DE_B
		ATRIBATIP_2_DE_B
		ATRIBATIP_3_DE_B
		B_AMB_ATRIBSTIPS
		ATRIBSATIPS_DE_B

INS2D:

5 Encaix: paral·lelogram oblic (no **NORM-XY**, però sí **NORM-Z**)

B_SENSE_ATRIBS	*B_333*	*ATRIBATIP_1_DE_B*
ATRIBS_DE_B		*ATRIBATIP_2_DE_B*
		ATRIBATIP_3_DE_B
		ATRIBATIP_1_DE_B_333
		ATRIBATIP_3_DE_B_798
		ATRIBSATIPS_DE_B_333

6 Encaix: rectangle (**NORM-XY** i **NORM-Z**)

		ATRIBATIP_1_DE_B
		ATRIBATIP_2_DE_B
		ATRIBATIP_3_DE_B
		B_AMB_ATRIBSTIPS
		ATRIBSATIPS_DE_B

A tall d'epíleg, acabarem amb les normes d'ús de les noves ordres que l'aplicació incorpora al repertori bàsic d'AutoCAD, tenint present que quan parlem d'atributs només ens referim als editables (és a dir, que en queden exclosos els Constants):

BLOQUEOK
Alternativa a ***BLOQUE*** o ***-BLOQUE***, garanteix que les insercions simples o múltiples generades per ***INS2D***/***INS3D*** i els objectes resultants de descompondre amb ***DESCOMPOK*** una inserció, seran correctes del tot. És imprescindible quan entre els components del bloc que volem crear hi ha atributs justificats amb l'opció ***M****edio* i el sistema de referència actual no és el ***SCU****niversal* (o no és paral·lel a ell ni el punt base del bloc coincideix amb l'origen); si no, aquests atributs ***M*** s'inseririen moguts. Quan detecta l'existència d'atributs justificats amb l'opció ***M****edio*, la canvia per l'opció ***M****edio*-***C****entro*: si els textos assignats estan formats per caràcters numèrics o lletres majúscules (menys "Ç"), exclusivament o barrejats amb lletres minúscules (tret de "ç", "g", "j", "p", "q" i "y"), la seva posició no variarà; altrament, es poden produir petits desplaçaments amunt o avall, raó per la qual serà preferible que els blocs que hagin de ser utilitzats per ***INS2D***/***INS3D*** evitin aquests atributs, així podrem usar ***BLOQUE***/***-BLOQUE*** tranquil·lament i no caldrà recórrer a ***BLOQUEOK***. ***BLOQUEOK*** actua com ***-BLOQUE***: en executar-la desapareixen els objectes seleccionats, però com ja passa amb aquesta ordre es poden recuperar tot seguit mitjançant ***UY***.

INS2D o ***INS3D***
És l'aportació fonamental d'aquest treball, que permet d'inserir un bloc de manera que 2 (***INS2D***) o 3 (***INS3D***) punts, situats en les direccions coordenades a partir del punt base, encaixin en unes posicions prefixades. Tret que aquestes posicions també descriguin direccions ortogonals (inserció simple), la inserció serà doble o triple (inserció d'una inserció prèvia, del bloc o d'una altra inserció prèvia). Quan la geometria del bloc sigui tridimensional, la diferència entre usar ***INS3D*** amb **O—Z** normal a **X—O—Y** o usar ***INS2D*** consisteix en què en el primer cas el factor d'escala $\mathbf{E_z}$ depèn de **O—Z** mentre que en el segon és $\mathbf{E_z = E_x}$ (això no vol dir que, si **Z** està situat a la vertical de **O** i a una distància **1**, hagi de ser **O—Z = O—X**). L'acció de ***INS2D***/***INS3D***, a banda d'aconseguir inserir el bloc encaixant-lo en els 3 o 4 punts especificats, supera algunes de les limitacions dels blocs d'AutoCAD:

- Situa correctament els atributs que no pertanyen al pla coordenat ***XY*** del bloc original (atributs atípics).

- Neutralitza el desplaçament enrera/endavant que experimenten els atributs quan no estan justificats sobre la línia base: allò que fa en realitat és substituir el seu punt base (situat a la línia superior, mitjana o inferior) per la seva projecció sobre aquesta línia.

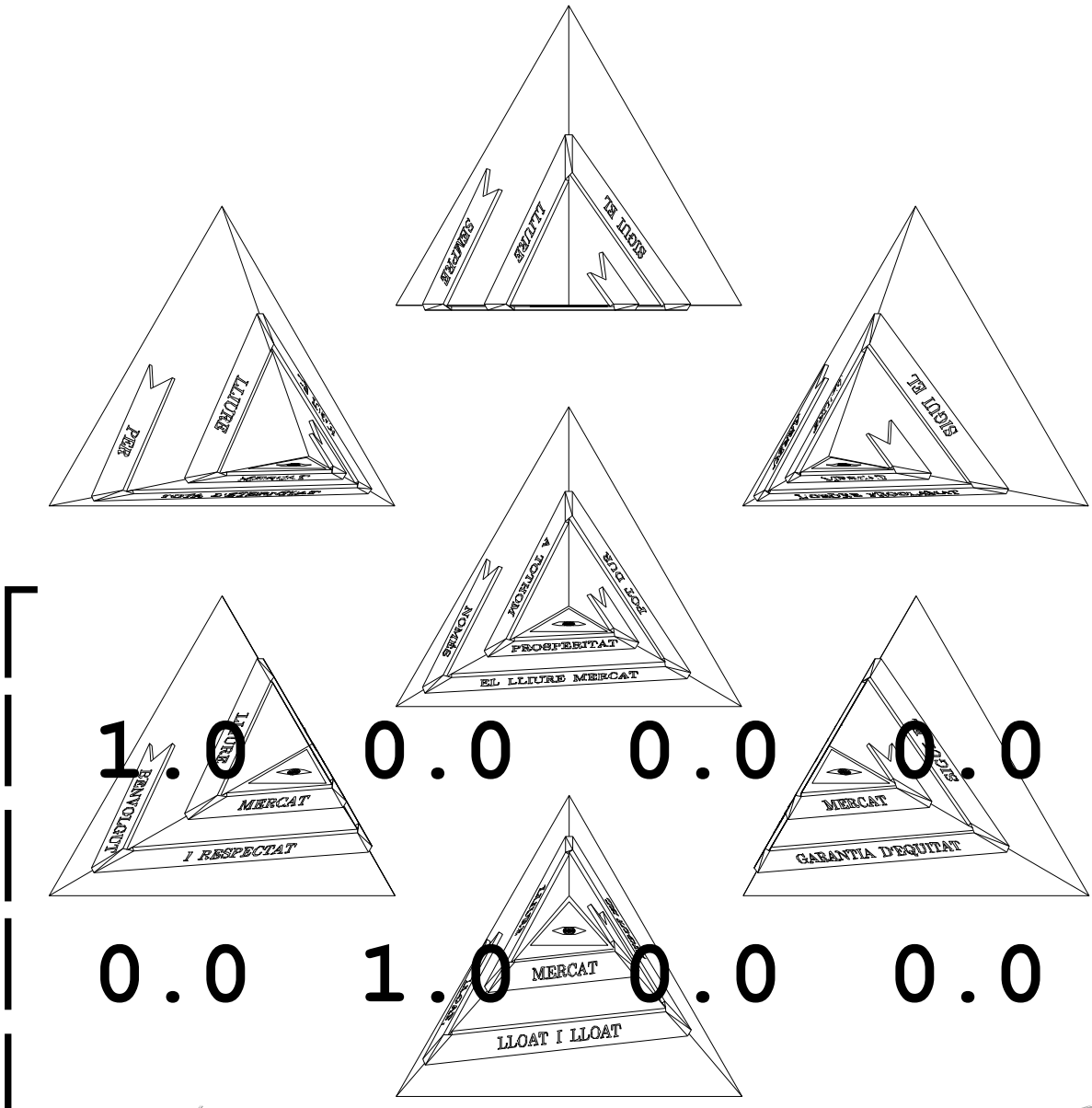
- Neutralitza l'aixafament transversal que experimenten els atributs escrits amb un estil de text definit amb caràcters inclinats, quan inserim el bloc portador amb escalat no uniforme i després s'explosiona.

INS2D o ***INS3D*** no manipulen directament el bloc original sinó uns blocs substituts (**NLOC-1** i **NLOC-2** en les insercions simples, **BLOC-1** i **BLOC-2** en les múltiples i, si s'escau, uns blocs portadors d'atributs atípics) i les abans esmentades insercions prèvies, convertides al seu torn en blocs (els anomenem blocs derivats genèrics, de primer i segon grau), que guardarem per si en un futur els necessitem. Tret del que ara direm de **BLOC-2** i **NLOC-2**, l'usuari no ha de manipular cap d'aquests blocs: si hem redefinit amb ***BLOQUE*** un bloc que prèviament havia estat utilitzat per ***INS2D*** o ***INS3D***, les futures aplicacions seguiran usant la definició antiga; la manera de forçar ***INS2D***/***INS3D*** a actualitzar-ne els continguts és eliminar amb ***LIMPIA*** el bloc auxiliar **BLOC-2** (el segon substitut de **BLOC** per a insercions obliqües, que en l'exemple de referència es deia *ATRIBS_DE_B*) o **NLOC-2** (el segon substitut per a insercions ortogonals, que en l'exemple de referència es deia *ATRIBSATIPS_DE_B*); abans no caldrà esborrar cap inserció perquè en el segon cas ja ho fa l'aplicació (si hi ha atributs atípics) o no arriba a usar-se (si no n'hi ha), i en el primer perquè totes les seves insercions estan explosionades. Després d'això, la primera vegada que reclamem el bloc amb ***INS2D*** o ***INS3D*** s'actualitzaran els blocs auxiliars existents i les seves insercions, amb les limitacions pròpies de les redefinicions fetes des de ***BLOQUE***, pel que fa als atributs.

DESCOMPOK

Per explosionar insercions (simples o múltiples) realitzades amb ***INS2D***/***INS3D***, és imprescindible si hi ha atributs atípics o que estiguin escrits amb un estil de text definit amb caràcters inclinats. Pel que fa a la propagació de l'explosió, té les mateixes limitacions que l'ordre ***DESCOMP***. Entre aquestes: quan el component és una inserció de bloc no es descompon automàticament però es pot explosionar de nou si l'escalat és uniforme o no està girada; altrament (factors d'escala diferents i angle de gir no nul) es converteix en inserció de bloc anònim ("*E1", "*E2", ...) no explosionable, i surten avisos "1 no se ha podido descomponer"; si l'aplicació ***INS2D***/***INS3D*** no és ortogonal (inserció múltiple) estem en aquest cas. A diferència de ***DESCOMP***, només permet de seleccionar una inserció cada vegada, però tant si fem clic a la part convencional de la inserció com en algun dels atributs atípics, la descomposició afecta al conjunt. A més de garantir el correcte posicionament dels atributs atípics i evitar l'aixafament transversal dels atributs amb escriptura obliqua, restituirà als atributs verificables el seu nom original (en explosionar una inserció amb atributs, aquests perden el valor assignat i el substitueixen per l'identificador) i es desempallegarà dels atributs invisibles (els identificadors sempre són visibles) incorporats per ***INS2D***/***INS3D***, com "BLOQUEOK" o "NO-BLOQUEOK", "WWAA-1" i "WWAA-2" (explosionant amb ***DESCOMP***, els dos últims només apareixerien en el cas d'una aplicació ***INS2D***/***INS3D*** ortogonal, és a dir, d'inserció simple).

Aquesta aplicació és compatible amb les versions 15 i 16 d'AutoCAD (denominacions comercials ACAD 2000 i 2002 d'una banda, i 2004 de l'altra), tot i haver-se trobat respostes específiques en algunes situacions, que hem consignat en aquest capítol i en el precedent, i que s'han resolt utilitzant com a condició de bifurcació del procés la variable **V>15** (**(> (atoi (substr (getvar "ACADVER") 1 2)) 15)**). Però cal afegir que, treballant amb ACAD 2004 i en més d'una ocasió (això sí, sempre usant blocs proveïts d'atributs atípics), algunes sessions han estat interrompudes per haver-se produït errors fatals inevitablement localitzats a la funció **INSERT*** i a causa de la fallida elevació al quadrat d'algun valor real. El primer diagnòstic va ser difícil, però el remei no podia ser més senzill: si anomenem **REAL** la base, n'hi va haver prou a substituir l'operació **(expt REAL 2)** per **(* REAL REAL)**. Ateses les circumstàncies, l'autor està convençut que les incidències s'han d'atribuir al fet d'estar fregant el límit de les prestacions del migrat equip físic amb què ha hagut de treballar, raó per la qual en el codi presentat aquí s'hi mantenen les expressions **(expt REAL 2)**. Però considera que convenia explicar-ho, per si de cas.



1.0 0.0 0.0 0.0

0.0 1.0 0.0 0.0

0.0 0.0 1.0 0.0

0.0 0.0 0.0 1.0

Consumidors compulsius de tot el món, uniu-vos!

Consumidors compulsius de tot el món, uniu-vos!

Consumidors compulsius de tot el món, uniu-vos!

Consumidors compulsius de tot el món, uniu-vos!

Consumidors compulsius de tot el món, uniu-vos!

Consumidors compulsius de tot el món, uniu-vos!

ANNEX: ENFOC ALGEBRAIC DE LES TRANSFORMACIONS AFINS

Aquest annex reprodueix bàsicament el contingut d'un text que va ser enviat per l'autor en 1986, tot just llegida la seva tesi doctoral, a la revista QÜESTIÓ de la UPC (no sé si avui desapareguda o rebatejada com a RESEARCH TRANSACTIONS) sense merèixer l'honor de ser publicada. Tot i concórrer les mateixes circumstàncies que en justificàren el refús (la principal, no incloure referències bibliogràfiques, segons l'informe de l'anònim *referee*), l'autor creu que la seva recuperació com a enfoc alternatiu a la línia aquí preconitzada (o complementari, si l'adoptem com una eina per a la verificació dels resultats) pot jugar de contrapunt interessant, malgrat que el seu propòsit més generalista no el faci directament confrontable amb un treball que orbita a l'entorn d'un aspecte concret (els blocs interns) d'un producte concret (AutoCAD). Obviat el títol primitiu, més xocant que il·lustratiu, s'han modificat detalls menors (l'ús d'un sistema de referència "mà esquerra" o la insistència de l'original en un aspecte ara allunyat del centre temàtic, com és la repercusió de les simetries en la caracterització dels elements de frontera dels models 3D). En acabat, s'ha cregut convenient de donar al lector l'oportunitat de provar el mètode postulat, posant a la seva disposició un *software* elemental però suficient per avaluar-ne la utilitat i, sobretot, per animar-lo a anar més enllà.

*** *** ***

En els processos de manipulació de formes geomètriques tridimensionals sovint es presenta la necessitat de transformar una figura d'acord amb uns imperatius de posicionament prèviament establerts, però sense que el camí per arribar-hi sigui evident. En aquest context, deduir la transformació indirectament, per la via d'una descomposició en accions simples i llur posterior síntesi, no sembla el procediment més adient, en comportar un esforç innecessari. El present article ofereix un mètode més immediat, que permet de definir tota transformació lineal a partir del mínim nombre de punts que la determinen.

1.- INTRODUCCIÓ

En algun moment de gairebé tota aplicació CAD cal recórrer a la transformació lineal d'una figura, per tal de tenir-la referida a un sistema coordenat diferent del propi, per col·locar-la en una posició en què determinada manipulació pugui resoldre's més còmodament o per integrar-la en el conjunt d'una escena. Segons la finalitat que es persegueixi, aquestes transformacions poden conservar la mètrica original o limitar-se a mantenir les relacions de paral·lelisme, però si deixem de banda el comportament geomètric i ens centrem a considerar només el sentit en què es planteja el problema, podrem establir dos tipus ben diferenciats d'aplicació:

- Les que es caracteritzen per definir les accions transformadores com a dades de partença que ens conduiran cap a un resultat desconegut d'entrada. Referim-nos, com a paradigma d'aquest plantejament, al cas d'un objecte que es mou d'acord amb lleis prefixades (per exemple, rotació a l'entorn d'un eix que es desplaça), del qual cal obtenir les successives posicions.

- Les que parteixen del coneixement previ dels efectes de la transformació, és a dir, de la figura transformada). A diferència de l'anterior, aquí el que imposem és el resultat final i allò que cal esbrinar és precisament el tipus d'acció que ens hi haurà de conduir.

El present article pretén posar en evidència l'escassa adequació dels algorismes emprats usualment al segon grup de problemes (estratègies bàsicment analítiques, mitjançant llur descomposició en parts fàcilment abordables) i proposar un mètode alternatiu, més globalitzador i d'orientació finalista.

La major part d'autors coincideixen a presentar tota transformació linial com a producte d'una seqüència de transformacions elementals, cadascuna referida a un tipus de moviment o acció fàcilment identificable. Si **P** és el conjunt de vèrtexs definidors de la figura original i **P′** el dels seus homòlegs a la transformada, utilitzant coordenades homogènies podríem expressar així una transformació 3D:

$$(P'_x, P'_y, P'_z, 1) = (P_x, P_y, P_z, 1)\ T$$

T és una matriu **4 × 4** que, segons quina sigui l'acció elemental considerada, incorpora les variacions següents sobre el patró identitat:

- Translació en **x**, **y** i **z**:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ D_x & D_y & D_z & 1 \end{bmatrix}$$

- Escalat en **x**, **y** i **z**:

$$T = \begin{bmatrix} A_x & 0 & 0 & 0 \\ 0 & B_y & 0 & 0 \\ 0 & 0 & C_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotacions al voltant dels eixos **X**, **Y** i **Z**:

$$T_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & B_y & B_z & 0 \\ 0 & C_y & C_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_y = \begin{bmatrix} A_x & 0 & A_z & 0 \\ 0 & 1 & 0 & 0 \\ C_x & 0 & C_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_z = \begin{bmatrix} A_x & A_y & 0 & 0 \\ B_x & B_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

On $A_x = B_y = C_z = \cos\theta$, $A_y = B_z = C_x = \sin\theta$ i $A_z = B_x = C_y = -\sin\theta$, si θ és l'angle girat. A diferència de translacions i escalats, el producte de diversos girs no és commutatiu, raó per la qual s'ha indivualitzat l'acció.

Sovint es parla de simetria i cisallament, però com que només són particularitats o derivacions de les accions anteriors (la primera no és més que una transformació d'escalat amb un o més elements negatius, mentre que la segona es pot resoldre com a combinació de girs i escalats), podem afirmar que qualsevol transformació linial es pot descompondre en una seqüència de translacions, girs i escalats, aplicades en un ordre precís. Tanmateix, tot i que aquesta estratègia modular és òptima per a aplicacions que compten amb una clara definició de les manipulacions que s'han de practicar, pot resultar feixuga en intentar traslladar-la a processos muntats sobre la consecució d'una resposta determinada.

Prenem com a mostra l'exemple de la Figura A.1, en què es tracta de transformar un cub, perfectament adaptat al sistema de referència, en un paral·lelepípede oblic i situat en una posició gens privilegiada en relació a aquest sistema. Coneixem la mètrica d'aquest cos (la situació dels punts P'_1, P'_2, P'_3 i P'_4) i volem trobar una transformació capaç de crear-lo a partir del cub primitiu, però amb els estris que hem enumerat només podem seguir un camí:

1) Dissenyar alguna seqüència d'accions elementals, T_1, T_2, ..., T_n, que acostin gradualment la figura transformada a l'original, fins que coincideixin.

2) Obtinguda la sèrie, com que
   $P = P'(T_1\ T_2 \ldots T_n)$, considerar que
   $P' = P\left(T_n^{-1} \ldots T_2^{-1}\ T_1^{-1}\right)$
   és a dir, que $T = T_n^{-1} \ldots T_2^{-1}\ T_1^{-1}$

La seqüència esmentada estarà integrada per:

T_1 - Translació en **x**, **y** i **z**, per dur P'_1 cap a P_1 (l'origen de coordenades).

T_2 - Gir al voltant de l'eix ***Z***, per situar $\overline{P'_1\ P'_2}$ sobre el pla ***ZX***.

T_3 - Gir al voltant de l'eix ***Y***, per situar $\overline{P'_1\ P'_2}$ sobre l'eix ***X***.

$\mathbf{T_4}$ - Gir al voltant de l'eix ***X***, per situar $\overline{\mathbf{P'_3\ P'_8}}$ sobre el pla ***XY***.

$\mathbf{T_5}$ - Escalat en **x**, **y**, **z**, per fer coincidir $\mathbf{P'_2}$ amb $\mathbf{P_2}$, alinear $\overline{\mathbf{P'_3\ P'_8}}$ amb $\overline{\mathbf{P_3\ P_8}}$ i situar $\mathbf{P'_4}$, $\mathbf{P'_5}$, $\mathbf{P'_6}$ i $\mathbf{P'_7}$ sobre el pla definit per $\mathbf{P_4}$, $\mathbf{P_5}$, $\mathbf{P_6}$ i $\mathbf{P_7}$.

$\mathbf{T_6}$ - Cisallament en **x**, segons **y**, per fer coincidir $\mathbf{P'_3}$ amb $\mathbf{P_3}$ i $\mathbf{P'_8}$ amb $\mathbf{P_8}$ ($\mathbf{B_x = -\frac{P'_{3x}}{P_{3y}}}$).

$\mathbf{T_7}$ - Cisallament en **y**, segons **z**, per alinear $\overline{\mathbf{P'_4\ P'_7}}$ amb $\overline{\mathbf{P_4\ P_7}}$ i $\overline{\mathbf{P'_5\ P'_6}}$ amb $\overline{\mathbf{P_5\ P_6}}$ ($\mathbf{C_y = -\frac{P'_{4y}}{P_{4z}}}$).

$\mathbf{T_8}$ - Cisallament en **x**, segons **z**, per fer coincidir $\mathbf{P'_4}$ amb $\mathbf{P_4}$, $\mathbf{P'_7}$ amb $\mathbf{P_7}$, $\mathbf{P'_5}$ amb $\mathbf{P_5}$ i $\mathbf{P'_6}$ amb $\mathbf{P_6}$ ($\mathbf{C_x = -\frac{P'_{4x}}{P_{4z}}}$).

Com el lector ja haurà intuït, en descriure cada transformació $\mathbf{T_m}$ ($\mathbf{m = 2, 3, \ldots 8}$) no utilitzem la notació $\mathbf{P'_n}$ ($\mathbf{n = 1, 2, 3, 4}$) amb el mateix significat que li havíem donat inicialment (en descriure $\mathbf{T_1}$) sinó amb la intenció de representar la posició que aquells punts van assolint al llarg del procés (és a dir, en descriure $\mathbf{T_{m-1}}$). Les tres últimes manipulacions es podrien reunir, sense interferències mútues, en una matriu única, però s'han individualitzat per a un millor seguiment del procés.

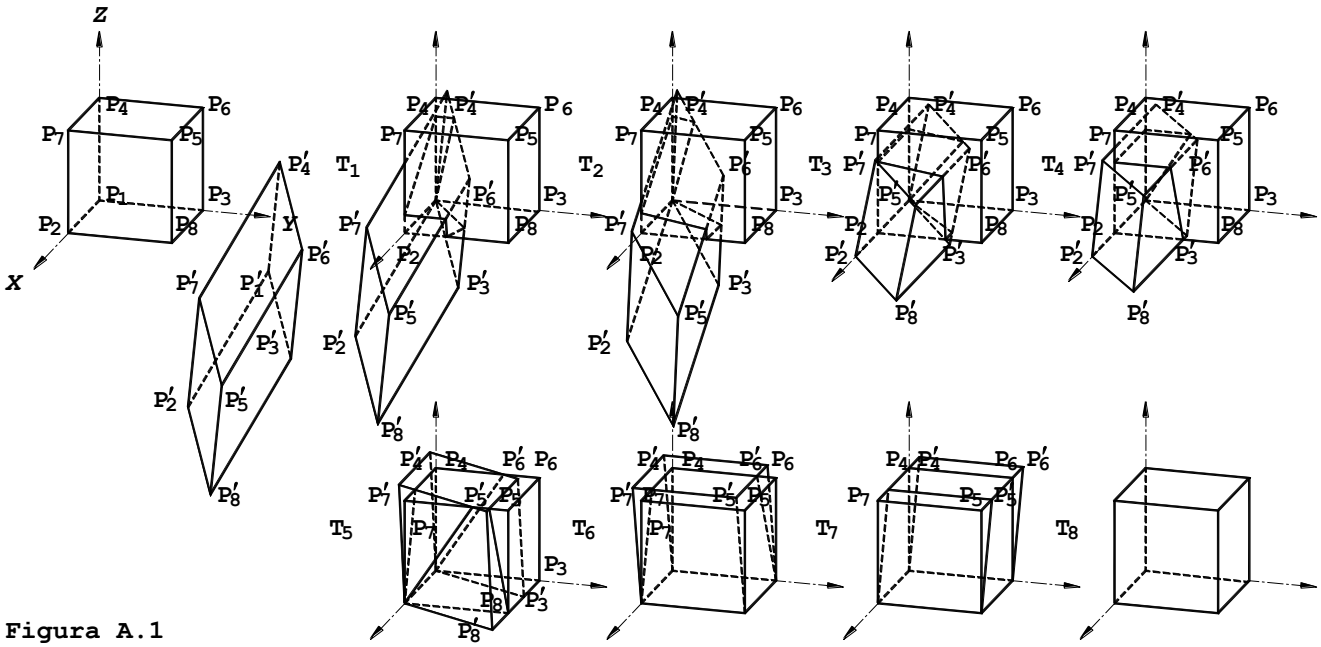


Figura A.1

D'aquest procediment se'n podrien dir moltes coses, llevat que sigui àgil: tot i que no presenta cap dificultat preparar subrutines orientades a la determinació de cadascuna de les transformacions, trobar quin és el camí que més directament porta de $\mathbf{P'}$ a $\mathbf{P}$, aplicar cada transformació als punts de referència per actualitzar-ne les posicions i plantejar la resolució de la següent, invertir-les totes (encara que això només consisteixi a canviar el signe dels elements significatius de les matrius de translació o escalat, o el de θ quan es tracta de gir) i obtenir-ne el producte, són massa petites operacions. Fins i tot quan la figura transformada fos un paral·lelepípede ortogonal (prescindir de $\mathbf{T_6}$, $\mathbf{T_7}$ i $\mathbf{T_8}$), inclús un cub d'aresta igual a l'original (prescindir de $\mathbf{T_5}$), la temptativa de defugir una estratègia tan perifràstica com poc generalitzable seria benvinguda.

El plantejament que des d'aquí proposem elimina aquesta visió particularista, en reduir el problema a la resolució d'un sistema d'equacions, on els paràmetres de la transformació (els elements de la matriu **T**) són les incògnites, i les dades (coordenades originals i transformades) juguen el paper de coeficients i termes independents: l'usuari només haurà d'intervenir per triar l'opció més econòmica, en el sentit de precisar el mínim nombre de punts que determinen la transformació, en funció del seu abast qualitatiu (translacions, girs i escalats; translacions i girs; translacions i escalats o només translacions).

2.- DEFINICIÓ PER 4 PARELLS DE PUNTS

En la seva formulació més general, tota transformació lineal **T** d'un punt **P** en **P′** es pot expressar:

$$P'_i = A_i P_x + B_i P_y + C_i P_z + D_i$$

El conjunt dels **12** valors A_i, B_i, C_i i D_i (**i** = **x, y, z**) només quedarà determinat si coneixem les coordenades de quatre punts no coplanaris i les dels transformats respectius, en formar un total de **12** equacions lineals (**n** = **1, 2, 3, 4**).

$$P'_i = A_i P_{nx} + B_i P_{ny} + C_i P_{nz} + D_i$$

representable matricialment (**P′ = P × T**) en la forma

$$\begin{bmatrix} P'_{1x} & P'_{1y} & P'_{1z} & 1 \\ P'_{2x} & P'_{2y} & P'_{2z} & 1 \\ P'_{3x} & P'_{3y} & P'_{3z} & 1 \\ P'_{4x} & P'_{4y} & P'_{4z} & 1 \end{bmatrix} = \begin{bmatrix} P_{1x} & P_{1y} & P_{1z} & 1 \\ P_{2x} & P_{2y} & P_{2z} & 1 \\ P_{3x} & P_{3y} & P_{3z} & 1 \\ P_{4x} & P_{4y} & P_{4z} & 1 \end{bmatrix} \begin{bmatrix} A_x & A_y & A_z & 0 \\ B_x & B_y & B_z & 0 \\ C_x & C_y & C_z & 0 \\ D_x & D_y & D_z & 1 \end{bmatrix} \qquad [1]$$

i que es descompon en **3** sistemes de **4** equacions, abordables de manera independent

$$\begin{aligned} P'_{nx} &= A_x P_{nx} + B_x P_{ny} + C_x P_{nz} + D_x \qquad (n = 1, 2, 3, 4) \\ P'_{ny} &= A_y P_{nx} + B_y P_{ny} + C_y P_{nz} + D_y \qquad " \\ P'_{nz} &= A_z P_{nx} + B_z P_{ny} + C_z P_{nz} + D_z \qquad " \end{aligned} \qquad [2]$$

Per facil·litar l'usuari la detecció d'errors fortuïts en les dades intruduïdes i donar-li l'opció a esmenar-les, caldrà comprovar que el determinant de la matriu **P** no és nul: en cas contrari s'hauria d'interrompre el procés i avisar-lo que els quatre punts originals se situen incorrectament en el mateix pla, que tres estan alineats o que n'hi ha algun de repetit. Superada aquesta primera condició, caldrà repetir l'operació amb el determinant de **P′**: si fos nul, això comportaria que la transformació ha degenerat en projecció, en aplicar l'espai 3D sobre el pla sobre-determinat per P'_1, P'_2, P'_3 i P'_4, raó per la qual també caldria tallar l'execució i advertir l'usuari d'aquesta circumstància.

Un cop obtinguts els paràmetres de la transformació i aplicada aquesta al conjunt dels vèrtexs que configuren l'objecte, encara restarà una comprovació: assegurar-se que no s'hagi produït cap inversió de sentit en l'orientació de les cares que delimiten el volum interior de l'espai exterior. Obrirem ara un parèntesi, per tal de copsar el significat que aquí es dóna a la paraula "inversió".

Si el model descriptiu de l'objecte no es limita a un entramat d'arestes sinó que recull informació relativa a les fronteres, caldrà representar les cares (aquest article es mou, en principi, en un context de fronteres polièdriques) de manera que l'ordre d'enumeració dels vèrtexs en tanqui el perímetre d'acord amb un sentit adoptat convencionalment i associat a la seva orientació com element de contenció del volum interior: així, quan l'observador miri frontalment la cara, acceptarem que la visualitza des de l'exterior si el seguiment del seu perímetre, d'acord amb la seqüència establerta en el model, descriu una rotació de sentit antihorari, o que la visualitza des de l'interior de l'objecte si el sentit del gir és horari. En el cos de la Figura A.2-a l'adopció d'aquest conveni ens donaria la descripció de les cares següent, vistes "des de fora": **1-3-8-9-2**, **1-2-7-4**, **9-10-11**, **1-4-6-3**, **4-7-11-10-5-6**, **3-6-5-8**, **8-5-10-9** i **9-11-7-2**.

Si apliquem a aquest volum la transformació definida pels quatre parells de punts

Fig. A.2-a		Fig. A.2-b
P_1 (0, 0, 0)	→	P'_1 (0, 0, 0)
P_2 (1, 0, 0)	→	P'_2 (1, 0, 0)
P_3 (0, 1, 0)	→	P'_3 (0,–1, 0)
P_4 (0, 0, 1)	→	P'_4 (0, 0, 1)

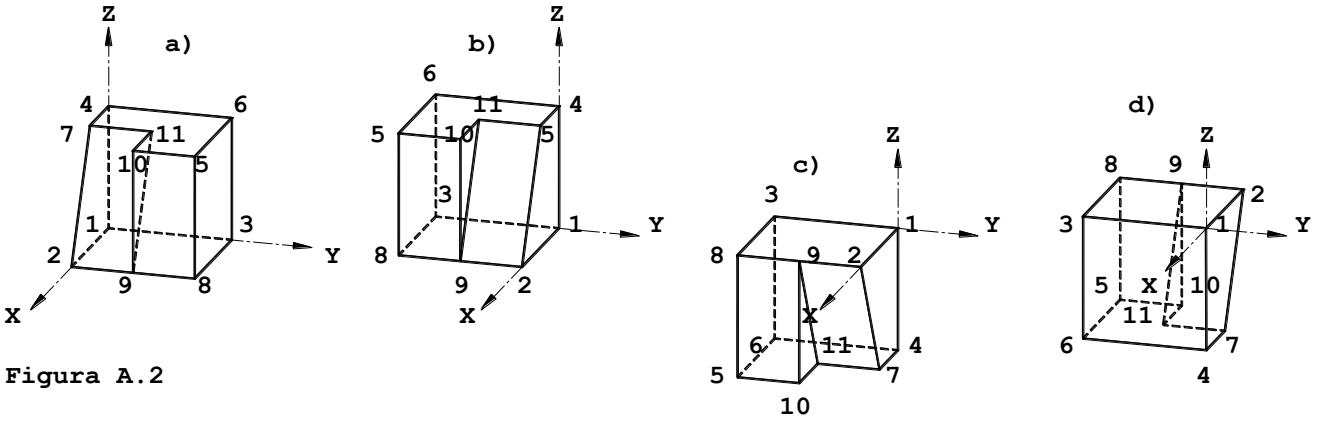


Figura A.2

la seva matriu coincidirà amb la identitat, tret de l'element $\mathbf{B_y = -1}$. Es tracta, doncs, d'una transformació d'escalat negatiu en **y**, i el resultat (Figura A.2-b) és un cos simètric respecte al pla ***ZX***. Això queda palès pel fet que ambdues figures no són congruents (no es poden superposar): encara que n'haguéssim considerat una amb múltiples simetries (un cub sencer, per exemple, sense la mossegada en forma de falca que n'ha trencat la regularitat) i la transformada fos morfològicament indiferenciable de l'original, no hi hauria manera de fer coincidir els vèrtexs homònims. Però per damunt de tot adonem-nos d'un aspecta de gran transcendència: si ara llegim el perímetre de les cares segons la descripció inicial, el sentit de gir que se'n deriva serà horari. Aquesta inversió de sentit no ens pot sorprendre, perque parlant de reflexió no fa més que reproduir una experiència ben familiar: en contemplar en el mirall la imatge d'un text, aquest esdevé illegible. Ara bé, darrera d'aquesta constatació s'hi amaga una conseqüència força més pertorbadora: la transformació haurà comportat un canvi en el caràcter material de la figura, que si abans representava un objecte situat en el buit ara caldria interpretar com un buit envoltat de matèria, si som conseqüents amb el conveni establert.

Ja veiem doncs que en determinades transformacions no n'hi haurà prou a substituir les coordenades dels vèrtexs de la figura per les transformades respectives, sinó que caldrà invertir l'ordre d'enumeració dels vèrtexs de cada polígon, per tal de restituir-la a la seva condició material primitiva. ¿Vol dir això que, sempre que en una transformació lineal intervinguin factors d'escala negatius s'hi introduirà subreptíciament una inversió material?: no sempre, i per discutir la incidència del fenòmen convé que seguim manipulant el cos de la Figura 2.

Definim ara dues noves transformacions:

Fig. A.2-a		Fig. A.2-c	Fig. A.2-a		Fig. A.2-d
$\mathbf{P_1\ (0,\ 0,\ 0)}$	→	$\mathbf{P'_1\ (0,\ 0,\ 0)}$	$\mathbf{P_1\ (0,\ 0,\ 0)}$	→	$\mathbf{P'_1\ (0,\ 0,\ 0)}$
$\mathbf{P_2\ (1,\ 0,\ 0)}$	→	$\mathbf{P'_2\ (1,\ 0,\ 0)}$	$\mathbf{P_2\ (1,\ 0,\ 0)}$	→	$\mathbf{P'_2\ (-1,\ 0,\ 0)}$
$\mathbf{P_3\ (0,\ 1,\ 0)}$	→	$\mathbf{P'_3\ (0,-1,\ 0)}$	$\mathbf{P_3\ (0,\ 1,\ 0)}$	→	$\mathbf{P'_3\ (0,-1,\ 0)}$
$\mathbf{P_4\ (0,\ 0,\ 1)}$	→	$\mathbf{P'_4\ (0,\ 0,-1)}$	$\mathbf{P_4\ (0,\ 0,\ 1)}$	→	$\mathbf{P'_4\ (0,\ 0,-1)}$

Pel que fa a la primera (segona, si comptem la d'abans), ja són dos els factors d'escala negatius, $\mathbf{B_y = C_z = -1}$, i malgrat això Fig. A.2-c conserva el caràcter material de Fig. A.2-a perquè l'orientació relativa de les cares no ha canviat. Cosa que no té res d'estrany, si considerem qualsevol d'aquests arguments:

- Els valors $\mathbf{B_y = C_z = -1 =} \cos \mathbf{180°}$ i $\mathbf{B_z = C_y = 0 =} \sin \mathbf{180°}$ poden interpretar-se també com a definidors d'un gir de **180°** al voltant de l'eix ***X***, moviment rígid sense simetria planària: si alguna simetria implícita tenim, és axial.
- També podem obtenir Fig. A.2-c a partir de Fig. A.2-b, en aplicar-hi un escalat negatiu en **z** (reflexió segons el pla ***XY***). Així, com que invertim allò que havia estat invertit prèviament, recobrarem l'orientació relativa de Fig. A.2-a, com es recupera la llegibilitat d'un text en la imatge deguda a la segona reflexió, sobre un joc de dos miralls perpendiculars.

Quant a l'última transformació, les diferències en relació a la matriu identitat s'hauran ampliat a $\mathbf{A_x = B_y = C_z = -1}$. El resultat serà el mateix que si haguéssim afegit un escalat negatiu en **x** a Fig. A.2-c (reflexió segons el pla ***YZ***), i tornarà a produir-se una inversió material: entre Fig. A.2-a i Fig. A.2-d ara tindrem una simetria central (o una homotècia amb raó de semblança **−1**, com vulgueu) respecte a l'origen de coordenades, comparable a la imatge deguda a la tercera reflexió en un joc de tres miralls ortogonals, de nou illegible si parlem d'un text.

No sembla gens dificil treure'n conclusions i preveure en quines circumstàncies una transformació provocarà la inversió del model de fronteres, obligant-nos a una nova inversió per tal que les cares recuperin l'orientació relativa primitiva:
1) Quan només inclogui un escalat negatiu (simetria planar), eventualment combinat amb translacions, girs o d'altres de positius, es produirà inversió.
2) No es produirà, en canvi, si apareixen dos escalats negatius (simetria axial), superposats o no a d'altres accions.
3) Si escalem negativament en totes tres direccions coordenades (simetria central) de nou hi haurà inversió.

Disortadament, el mètode que proposem no ens permet d'utilitzar aquests criteris de forma immediata. La dificultat rau en el fet que, a diferència de la tècnica analítica, la presència d'escalats negatius no és palès per a l'usuari, que fixa les condicions d'arribada però desconeix el camí. D'altra banda, fóra precipitat resumir aquestes conclusions en una fàcil recepta que únicament parés esment en la paritat del nombre d'elements negatius a la diagonal principal de la matriu de transformació: afirmar que es produeix inversió material quan **1** o **3** dels factors d'escala ($\mathbf{A_x}$, $\mathbf{B_y}$, $\mathbf{C_z}$) són negatius, i només en aquest cas, seria caure en un parany probablement derivat del simplisme dels exemples concrets utilitzats a la Figura A.2, on les transformacions eren escalats purs. Aquesta norma encara seria vàlida si també hi consideréssim translacions, però en intervenir-hi girs tot se n'aniria en orris: tant pot ser que a la diagonal principal hi aparegui un nombre senar d'elements negatius i que no hi hagi inversió, com que no es doni aquesta circumstància i n'hi hagi. Haurem de recórrer a un criteri més ampli i que estigui directament emparentat amb el conveni d'orientació de les cares.

Per què afirmàvem que es produïa inversió material, en les Figures A.2-b i A.2-d? Doncs perquè a la Figura primitiva A.2-a, posem per cas, el gir al voltant de $\overline{\mathbf{P_1\,P_2}}$ que du a la semirecta $\overline{\mathbf{P_1\,P_3}}$ a coincidir amb $\overline{\mathbf{P_1\,P_4}}$ té sentit horari si l'observem des de $\mathbf{P_2}$, mentre que serà antihorari en repetir l'observació en qualsevol de les transformades esmentades. La traducció algebraica d'aquesta experiència no pot ser més senzilla: com que el producte vectorial $\overline{\mathbf{P_1\,P_3}} \times \overline{\mathbf{P_1\,P_4}}$ és un vector normal al pla definit per $\mathbf{P_1}$, $\mathbf{P_3}$ i $\mathbf{P_4}$, l'orientació del qual està lligada a l'esmentat sentit de gir, i el signe del producte escalar entre aquest i el vector $\overline{\mathbf{P_1\,P_2}}$ ens indica si ambdós vectors es troben o no en el mateix semiespai (a la mateixa banda del pla), n'hi haurà prou a calcular el producte mixt de $\overline{\mathbf{P_1\,P_2}}$, $\overline{\mathbf{P_1\,P_3}}$ i $\overline{\mathbf{P_1\,P_4}}$ (és a dir, el producte escalar entre $\overline{\mathbf{P_1\,P_2}}$ i el resultat del producte vectorial $\overline{\mathbf{P_1\,P_3}} \times \overline{\mathbf{P_1\,P_4}}$, i el dels transformats respectius. Només si veiem que tenen diferent signe (cap dels dos productes pot ser nul, atès que la quaterna **P** no és coplanària, ni tampoc **P′**) podrem assegurar que hi ha hagut inversió. En definitiva, quan

$$\overline{\mathbf{P_1\,P_2}} \cdot \left(\overline{\mathbf{P_1\,P_3}} \times \overline{\mathbf{P_1\,P_4}}\right) = \begin{vmatrix} (\mathbf{P_{2x}} - \mathbf{P_{1x}}) & (\mathbf{P_{2y}} - \mathbf{P_{1y}}) & (\mathbf{P_{2z}} - \mathbf{P_{1z}}) \\ (\mathbf{P_{3x}} - \mathbf{P_{1x}}) & (\mathbf{P_{3y}} - \mathbf{P_{1y}}) & (\mathbf{P_{3z}} - \mathbf{P_{1z}}) \\ (\mathbf{P_{4x}} - \mathbf{P_{1x}}) & (\mathbf{P_{4y}} - \mathbf{P_{1y}}) & (\mathbf{P_{4z}} - \mathbf{P_{1z}}) \end{vmatrix} \qquad [3]$$

tingui diferent signe que

$$\overline{\mathbf{P'_1\,P'_2}} \cdot \left(\overline{\mathbf{P'_1\,P'_3}} \times \overline{\mathbf{P'_1\,P'_4}}\right) = \begin{vmatrix} (\mathbf{P'_{2x}} - \mathbf{P'_{1x}}) & (\mathbf{P'_{2y}} - \mathbf{P'_{1y}}) & (\mathbf{P'_{2z}} - \mathbf{P'_{1z}}) \\ (\mathbf{P'_{3x}} - \mathbf{P'_{1x}}) & (\mathbf{P'_{3y}} - \mathbf{P'_{1y}}) & (\mathbf{P'_{3z}} - \mathbf{P'_{1z}}) \\ (\mathbf{P'_{4x}} - \mathbf{P'_{1x}}) & (\mathbf{P'_{4y}} - \mathbf{P'_{1y}}) & (\mathbf{P'_{4z}} - \mathbf{P'_{1z}}) \end{vmatrix} \qquad [4]$$

i només en aquest cas, l'orientació relativa de les cares s'haurà invertit a cavall de la transformació, i caldrà redefinir aquesta (canviant adequadament algun $\mathbf{P'_i}$) o forçar una nova inversió per recuperar la primitiva qualificació material de la figura.

Com que tant se'ns dóna de realitzar la correcció (sigui redefinint **T** o invertint-ne el resultat) abans o després d'haver obtingut els paràmetres $\mathbf{A_i}$, $\mathbf{B_i}$, $\mathbf{C_i}$ i $\mathbf{D_i}$, serà millor que esperem a disposar d'aquests valors. Així, en comptes de calcular dos productes mixtos n'hi haurà prou a conèixer el signe del producte mixt únic

$$\overline{\mathbf{A}} \cdot \left(\overline{\mathbf{B}} \times \overline{\mathbf{C}}\right) = \begin{vmatrix} \mathbf{A_x} & \mathbf{A_y} & \mathbf{A_z} \\ \mathbf{B_x} & \mathbf{B_y} & \mathbf{B_z} \\ \mathbf{C_x} & \mathbf{C_y} & \mathbf{C_z} \end{vmatrix} \qquad [5]$$

i actuar segons que sigui:

- $\bar{A} \cdot (\bar{B} \times \bar{C}) > 0$ → no s'ha produït inversió
- $\bar{A} \cdot (\bar{B} \times \bar{C}) < 0$ → s'ha produït inversió → cal redefinir **T** o nova inversió.

En efecte, si prenem els punts $P_1\ (0, 0, 0)$, $P_2\ (1, 0, 0)$, $P_3\ (0, 1, 0)$ i $P_4\ (0, 0, 1)$, els seus transformats segons **[1]** seran

$P'_1 \equiv (D_x, D_y, D_z)$

$P'_2 \equiv (A_x + D_x,\ A_y + D_y,\ A_z + D_z)$

$P'_3 \equiv (B_x + D_x,\ B_y + D_y,\ B_z + D_z)$

$P'_4 \equiv (C_x + D_x,\ C_y + D_y,\ C_z + D_z)$

valors que, en ser substituïts a **[3]** i **[4]**, donen **1** i el producte **[5]** respectivament.

3.- DEFINICIÓ PER 3 PARELLS DE PUNTS

Si sabem d'entrada que la transformació és una isometria (conserva les distàncies entre punts, de manera que es pot entendre com un moviment rígid on hi considerem translacions i girs, limitant els escalats als casos $|A_x| = |B_y| = |C_z| = 1$), podríem definir-la amb tres punts no alineats i els seus transformats, per bé que aquests haurien de complir $P'_1{-}P'_2 = P_1{-}P_2$, $P'_1{-}P'_3 = P_1{-}P_3$ i $P'_2{-}P'_3 = P_2{-}P_3$. Tanmateix, perquè la introducció de dades sigui còmoda, sense que l'usuari s'hagi de preocupar d'aquest requisit, organitzarem el joc en els termes següents:

1) Introducció de P_1 i el seu transformat P'_1.

2) Introducció de P_2 i d'un punt Q' amb el qual la semirecta $\overline{P'_1\,Q'}$ sigui homòloga de la $\overline{P_1\,P_2}$. El programa ja cercarà l'homòleg de P_2, P'_2, que haurà de pertànyer a la primera semirecta tot respectant la distància $P'_1{-}P'_2 = P_1{-}P_2$. D'acord amb aquest plantejament, serà ($i = x, y, z$):

$$P'_{2i} = P'_{1i} + (Q'_i - P'_{1i}) \sqrt{\frac{(P_{2x} - P_{1x})^2 + (P_{2y} - P_{1y})^2 + (P_{2z} - P_{1z})^2}{(Q'_x - P'_{1x})^2 + (Q'_y - P'_{1y})^2 + (Q'_z - P'_{1z})^2}} \qquad [6]$$

3) Introducció d'uns punts, **R** (no alineat amb P_1 i P_2) i S' (no alineat amb P'_1 i Q'), amb els quals el semiplà definit per la recta $\overline{P'_1\,Q'}$ i S' sigui transformat del que determina $\overline{P_1\,P_2}$ amb **R**. A partir d'aquestes dades, el programa podría trobar un punt R' del primer semiplà que, en guardar les distàncies $P'_1{-}R' = P_1{-}R$ i $P'_2{-}R' = P_2{-}R$, seria el transformat de **R**. Però el càlcul resultarà més senzill si, en comptes de **R** i com a homòlegs, localitza uns punts P_3 i P'_3 en què els vectors $\overline{P_1\,P_3}$ i $\overline{P'_1\,P'_3}$ siguin normals als plans definits per P_1, P_2, **R** i per P'_1, P'_2, S', respectivament, i que tinguin el mateix mòdul (és a dir, que respectin les distàncies $P'_1{-}P'_3 = P_1{-}P_3$ i $P'_2{-}P'_3 = P_2{-}P_3$). D'acord amb aquesta opció, obtindrem els productes vectorials $\bar{U} = \overline{P_1\,P_2} \times \overline{P_1\,R}$ i $\bar{V}' = \overline{P'_1\,P'_2} \times \overline{P'_1\,S'}$:

$$U_i = (P_{2j} - P_{1j})(R_k - P_{1k}) - (P_{2k} - P_{1k})(R_j - P_{1j}) \qquad [7]$$

$$V'_i = (P'_{2j} - P'_{1j})(S'_k - P'_{1k}) - (P'_{2k} - P'_{1k})(S'_j - P'_{1j}) \qquad (ijk = xyz,\ yzx,\ zxy) \qquad [8]$$

Naturalment, si $U_x = U_y = U_z = 0$ (o si $V'_x = V'_y = V'_z = 0$) caldrà interrompre el procés i alertar l'usuari, en estar **R** alineat amb P_1 i P_2 (o S' alineat amb P'_1 i Q'). Ja amb el vector $\bar{U}$, només caldrà aplicar-lo sobre P_1 per localitzar P_3:

$$P_{3i} = P_{1i} + U_i \qquad (i = x, y, z) \qquad [9]$$

Pel que fa a P'_3, després d'aplicar $\bar{V}'$ sobre P'_1 caldrà modificar-ne la longitud per tal que el seu mòdul sigui igual al de $\bar{U}$:

$$P'_{3i} = P'_{1i} + V'_i \sqrt{\frac{U_x^2 + U_y^2 + U_z^2}{V'^2_x + V'^2_y + V'^2_z}} \qquad (i = x, y, z) \qquad [10]$$

4) Encara que amb la introducció dels parells de punts P_1, P'_1 ($P_1 \rightarrow P'_1$), P_2, Q ($P_2 \rightarrow P'_2$) i R, S' ($P_3 \rightarrow P'_3$) la transformació isomètrica queda perfectament determinada, només tenim **9** equacions (**4 × 4** coordenades de punts transformats) per obtenir els **12** paràmetres. Podríem completar el sistema **[2]** (ara reduït a $n = 1, 2, 3$) amb les equacions

$$A_x^2 + A_y^2 + A_z^2 = 1$$

$$B_x^2 + B_y^2 + B_z^2 = 1$$

$$C_x^2 + C_y^2 + C_z^2 = 1$$

que surten d'usar **[6]** i **[7]** com a punts de referència (originals i transformats) i d'imposar-los la condició que es mantinguin les distàncies $P'_1{-}P'_2 = P_1{-}P_2 = 1$, $P'_1{-}P'_3 = P_1{-}P_3 = 1$, $P'_1{-}P'_4 = P_1{-}P_4 = 1$. Però és més senzill aprofitar el dispositiu ja implantat en l'apartat precedent (resolució general dels tres sistemes **[2]**), considerant un quart parell de punts homòlegs, de fàcil obtenció a partir dels tres que ja tenim. Adoptarem P_4 i P'_4 en què el vector $\overline{P_1\,P_4}$ sigui normal al pla P_1, P_2, P_3 (pertany al pla P_1, P_2, R), el vector $\overline{P'_1\,P'_4}$ ho sigui al pla P'_1, P'_2, P'_3 (pertany al pla P'_1, P'_2, S') i ambdós tinguin el mateix mòdul. N'hi ha prou a obtenir els productes vectorials $\overline{P_1\,P_2} \times \overline{P_1\,P_3}$ i $\overline{P'_1\,P'_2} \times \overline{P'_1\,P'_3}$, aplicant-los sobre els punts P_1 i P'_1 respectivament. Així,

$$P_{4_i} = P_{1i} + (P_{2j} - P_{1j})(P_{3k} - P_{1k}) - (P_{2k} - P_{1k})(P_{3j} - P_{1j}) \qquad [11]$$

$$P'_{4_i} = P'_{1i} + (P'_{2j} - P'_{1j})(P'_{3k} - P'_{1k}) - (P'_{2k} - P'_{1k})(P'_{3j} - P'_{1j}) \quad (ijk = xyz, yzx, zxy) \qquad [12]$$

El manteniment de l'orientació relativa de les cares quedarà assegurada de manera automàtica per aquesta definició de P_4 i P'_4, que s'escapa al control de l'usuari, raó per la qual podrem prescindir de la discussió del signe de **[5]**, preceptiva a l'apartat precedent (determinació per quatre parells de punts, de lliure elecció). Pel que fa a l'introducció de P_2, Q, R i S', hem d'insistir en la necessitat de respectar escrupolosament les regles del joc: quan a 2 i 3 parlàvem de semirectes i semiplans (i no simplement de rectes i plans), volíem dir que Q' s'ha de prendre a la banda de P'_1 on volem que se situï P'_2, homòleg de P_2, i que cal col·locar S' al costat de $\overline{P'_1\,P'_2}$ on esperem tenir R', homòleg de R (tal i com hem realitzat els productes **[13]** i **[14]**, P_4 ocuparà el semiplà oposat a R, i P'_4 l'oposat a R', amb una disposició equivalent a l'anterior).

4.- DEFINICIÓ PER 2 PARELLS DE PUNTS

Si en la transformació només hi intervenen translacions i escalats, serà possible descriure-la amb dos parells de punts, sempre que les rectes determinades per cada parella no siguin paral·leles a algun dels plans coordenats, és a dir, sempre que $P_{1i} \neq P_{2i}$ i $P'_{1i} \neq P'_{2i}$ ($i = x, y, z$).

En principi això voldria dir que, si la figura primitiva fos un paral·lelepípede ortogonal (orientat segons les direccions coordenades), no podríem utilitzar com a referència cap aresta ni cap diagonal de les cares: caldria recórrer a dos punts que no compartissin la mateixa cara (si es tractés de vèrtexs, haurien de ser dels que delimiten les diagonals del paral·lelepípede). Tot i així, més endavant veurem en quins casos seria possible de tolerar aquestes entrades, acceptant determinades hipòtesis.

A diferència dels apartats anteriors no haurem de formar tres sistemes de quatre equacions ja que, en no haver-hi rotacions, serà

$$A_y = A_z = B_x = B_z = C_x = C_y = 0 \qquad [13]$$

i, per trobar els altres **6** elements significatius de T (cap dels de la diagonal principal no podrà ser nul), n'hi haurà prou a resoldre aquests **3** sistemes de **2** equacions:

$$P'_{nx} = A_x P_{nx} + D_x \qquad (n = 1, 2)$$
$$P'_{ny} = B_y P_{ny} + D_y \qquad "$$
$$P'_{nz} = C_z P_{nz} + D_z \qquad "$$

Tindrem, doncs:

$$A_x = \frac{P'_{2x} - P'_{1x}}{P_{2x} - P_{1x}} \qquad B_y = \frac{P'_{2y} - P'_{1y}}{P_{2y} - P_{1y}} \qquad C_z = \frac{P'_{2z} - P'_{1z}}{P_{2z} - P_{1z}} \qquad [14]$$

$$D_i = \frac{P_{2i}\, P'_{1i} - P_{1i}\, P'_{2i}}{P_{2i} - P_{1i}} \qquad (i = x, y, z) \qquad [15]$$

Tal i com havíem fet en els dos apartats precedents, per evitar indeterminacions o incompatibilitats, abans caldrà detectar possibles errors a les dades introduïdes. Però, a diferència dels casos que allà contemplàvem, ara convindrà filar més prim per tal de flexibilitzar la introducció de dades. Així, quan

$P_{1i} = P_{2i}$ però $P'_{1i} \neq P'_{2i}$, o bé quan $P_{1i} \neq P_{2i}$ i $P'_{1i} = P'_{2i}$ (per a algun $i = x, y, z$) haurem d'alertar l'usuari i donar-li l'ocasió de rectificar P_1 o P_2: en el primer cas el sistema no tindria solució, i en el segon es tractaria d'una transformació degenerada (tota la figura es projectaria sobre el pla $i = P'_{2i}$). Tanmateix, quan

$P_{1i} = P_{2i}$ i $P'_{1i} = P'_{2i}$ (per a algun $i = x, y, z$), podríem suposar que

$$i = x \rightarrow A_x = 1 \qquad i = y \rightarrow B_y = 1 \qquad i = z \rightarrow C_z = 1 \qquad [16]$$

i així hauríem obviat la indeterminació: si l'usuari va escollir segments $\overline{P_1 P_2}$ i $\overline{P'_1 P'_2}$ paral·lels a un mateix pla coordenat, deixant sense definir l'escalat segons l'eix normal, tàcitament estava acceptant que en aquesta direcció no n'hi volia, d'escalat. Naturalment, de la hipòtesis adoptada es deduiría que

$$D_i = P'_{1i} - P_{1i} \qquad (i = x, y, z) \qquad [17]$$

Quant a la possibilitat que s'inverteixi l'orientació de les cares de l'objecte, la seva detecció ja no presenta la complexitat de l'apartat 2: en no haver-hi cap gir que afecti la diagonal principal de la matriu **T**, el primer pla d'actuació que allà havíem traçat (i que vam haver de refusar) aquí serà perfectament aplicable; en conseqüència, podem afirmar que, quan hi hagi un nombre senar d'elements A_x, B_y, C_z negatius, la transformació comportarà simetria. És més: com que cap dels tres valors és nul, n'hi haurà prou a referir-nos al signe del producte mixt

$$\bar{A} \cdot \left(\bar{B} \times \bar{C}\right) = A_x B_y C_z \qquad [18]$$

5.- DEFINICIÓ PER 1 PARELL DE PUNTS

Amb només P_1 i el seu homòleg P'_1 són dos els tipus de transformació que podríem definir: escalat pur o translació pura. Davant l'alternativa, sembla més raonable decidir-se per la segona via, perquè en la pràctica del modelatge de sòlids seria de dubtosa utilitat una transformació que sols donés opció a escalar la primitiva sobre el seu propi sistema de referència, normalment triat en funció d'una còmoda quantificació de les coordenades dels vèrtexs: escalant-la respecte a una altra posició ja necessitaríem definir-la amb dos parells de punts i ens situaríem en l'últim cas tractat (translació, escalat i translació). Fent-ho en relació a uns altres eixos caldria anar a quatre parells de punts i tornaríem al primer cas, si bé és cert que podríem haver ampliat el menú i desdoblar el segon (3), considerant no sols translacions i girs sinó també escalat uniforme: sol·licitaríem P'_2 en lloc de Q', evitaríem **[6]**, multiplicaríem **[10]** per $\frac{P'_1 - P'_2}{P_1 - P_2}$, i **[11]** i **[12]** per l'invers $\frac{P_1 - P_2}{P'_1 - P'_2}$ (en el quadre sinòptic que oferirem tot seguit hi apreciareu aquest desdoblament). Si ens decidim, doncs, per la variant translacional, a més d'acomplir-se **[15]** serà:

$$A_x = B_y = C_z = 1 \qquad [19]$$
$$D_i = P'_{1i} - P_{1i} \qquad (i = x, y, z) \qquad [20]$$

Òbviament, en tractar-se d'un cas particular de la família de transformacions contemplada a l'apartat 3, mai no pot produir-se la temuda inversió material.

Parells de punts	**1**	**2**	**3**		**4**
Accions simultànies	translació	translació escalat	translació rotació	translació rotació escalat unif.	translació rotació escalat
Introducció de dades	$P_1 \rightarrow P'_1$	$P_1 \rightarrow P'_1$ $P_2 \rightarrow P'_2$	$P_1 \rightarrow P'_1$ $P_2 - Q'$ $\quad P_2 \rightarrow P'_2$ (P'_2 **[6]**) $R - S'$		$P_1 \rightarrow P'_1$ $P_2 \rightarrow P'_2$ $P_3 \rightarrow P'_3$ $P_4 \rightarrow P'_4$
Control de dades (nova introducció)	-	$P_{1i} = P_{2i}$ i $P'_{1i} \neq P'_{2i}$ o $P_{1i} \neq P_{2i}$ i $P'_{1i} = P'_{2i}$ (en algun $i = x, y, z$)	$U_x = U_y = U_z = 0$ i/o $V'_x = V'_y = V'_z = 0$ (veieu en **[9]** i **[10]**)	determ. $P = 0$ i/o determ. $P' = 0$ (veieu en **[1]**)	
Elaboració	-	-	P_3 **[9]**, P'_3 **[10]** P_4 **[11]**, P'_4 **[12]** $P'_3 \leftarrow P'_3 \frac{P'_1 - P'_2}{P_1 - P_2}$		-
Determinació de la transformació	**[13]**, **[19]**, **[20]**	$P_{1i} \neq P_{2i}$ i $P'_{1i} \neq P'_{2i} \rightarrow$ $\rightarrow$ **[13]**, **[14]**, **[15]** $P_{1i} = P_{2i}$ i $P'_{1i} = P'_{2i} \rightarrow$ $\rightarrow$ **[13]**, **[16]**, **[17]**	resolució dels **3** sistemes **[2]**		
Inversió dels perímetres	-	**[18] < 0**	-		**[5] < 0**

5.- CONCLUSIONS

El sistema proposat, que resumim en el quadre annex, ens allibera indubtablement de l'obligació de descompondre el problema en etapes a la mida d'un instrumental força eficient a l'hora de deduir l'efecte a partir de la causa, però que no ho és gens si la qüestió es planteja a l'inrevés. Així i tot, seria legítim de mantenir alguna reticència en relació a la pretesa rendibilitat del mètode:

- Es podrà objectar que el supòsit pràctic que el justifica (fixar *a priori* els resultats) no té massa a veure amb la problemàtica real de l'usuari-dissenyador. Efectivament, si el component heurístic d'un procés de disseny és important, per respondre a l'estratègia del "provem de fer això i vejam què passa ..." (típica de la fase inicial d'ideació d'una forma), l'objecció serà vàlida. Però quan ja s'hagin assolit conclusions satisfactòries i calgui materialitzar els esborranys succesius en una geometria precisa, convé disposar d'una eina adient: si estem modelant un volum i pretenem que totes les peces encaixin al seu lloc, movent cada nou mòdul per adaptar-lo i soldar-lo a la part ja constituïda, la mètrica d'aquesta ultima serà la que manarà.
- En relació a la transformació d'un cub en paral·lelepípede oblic (Figura A.1), adoptada com a exemple en la introducció, pot adduir-se que, si per posicionar **8** punts (els vèrtexs P'_1 ... P'_8 de la figura transformada) necessitem explicitar-ne **4** i **4** més de referència (P'_1 ... P'_4 i P_1 ... P_4), no es veu el negoci per enlloc. Però l'argument seria trampós, perquè no és el mateix basar-se en punts coneguts que deduir les coordenades d'unes posicions previsibles però no quantificades i perquè no sempre treballarem amb volums tan elementals (només cal pensar en els **32** vèrtexs d'un cilindre aproximat matusserament a prisma amb la base poligonal de **16** costats).
- Tot i així, la perspectiva d'introduir manualment **24** valors numèrics per cada transformació afí ($2 \times 4 \times 3$ coordenades) no deixa de ser preocupant. Això és ben cert, però si l'únic canal d'accés és el teclat perquè un encara no disposa de perifèrics gràfics d'entrada de dades*, la insuficiència serà imputable a les limitacions de l'equip i no exclusivament a l'estratègia operativa: si implantem subrutines orientades a codificar les posicions marcades amb un llapis òptic sobre la imatge de la part de figura ja constituïda, el problema quedarà resolt, almenys pel que fa als "punts de contacte".

* L'observació pot semblar ingènua, però en 1986 tot just començaven a populalitzar-se les ratetes.

*** *** ***

Havent completat la transcripció de l'article nonat a què ens referíem a l'inici d'aquest annex, seguirem amb el també anunciat subministrament d'un *software* que ens permetrà de posar en solfa el mètode que s'hi proposa. Com que només es tracta de sortir del pas sense massa refinaments (ja els farà el lector, si l'interessa), en comptes de partir de zero desempolsarem unes rutines que poden ser-nos útils:

```
; Multiplica la matriu A (NIxNK) per la matriu B (NKxNJ).
; Les matrius han de representar-se com a llistes de subllistes-fila; els vectors,
; tant si són v-fila (A) com v-columna (B), poden introduir-se com a llistes
; simples (la funció ja els formatejarà adequadament).
(defun AB (A B / C AxB AIxB AIxBJ NI NJ NK I J K)
  (if (atom (car A)) (setq A (list A)))
  (if (atom (car B)) (progn
                       (foreach E B (setq C (append C (list (list E)))))
                       (setq B C)))
  (if (= (setq NK (length (car A))) (length B))
    (progn
      (setq NI (length A) NJ (length (car B)) I -1)
      (repeat NI
        (setq I (1+ I) J -1 AIxB ())
        (repeat NJ
          (setq J (1+ J) K -1 AIxBJ 0)
          (repeat NK (setq K (1+ K) AIxBJ (+ AIxBJ (* (nth K (nth I A))
                                                      (nth J (nth K B))))))
          (setq AIxB (append AIxB (list AIxBJ))))
        (setq AxB (append AxB (list AIxB)))))))

(defun TRANSPOSAR (A / LLE LL LE E LS LLS)
  (setq LLE A)
  (while LLE
         (setq LS () LL LLE)
         (foreach L LL
                  (setq E (car L) LE (cdr L)
                        LS (append LS (list E))
                        LLE (if (not (equal LLE LL)) LLE)
                        LLE (if LE (append LLE (list LE)))))
         (setq LLS (append LLS (list LS)))))

; Adapta el tercer element A de NENTSEL al format del tercer element de NENTSELP,
; requerit pel segon argument (optatiu) de GRVECS. Altres exigències de GRVECS són
; que SCP=SCU i que el primer argument respongui a una inserció neutra a l'origen.
(defun ADAPSEL (A) (append (TRANSPOSAR A) (list '(0 0 0 1))))

(defun DET (LL / M N LON SIG TOT MENOR L*)
  (if (= 2 (setq LON (length LL)))
      (- (* (car (car LL))(cadr (cadr LL))) (* (car (cadr LL))(cadr (car LL))))
      (progn
        (setq TOT 0 SIG -1 N -1)
        (foreach E1 (car LL)
          (setq SIG (- SIG) N (1+ N) MENOR ()
                MENOR (foreach L (cdr LL)
                        (setq M LON L* ()
                              MENOR (append MENOR
                                      (list (foreach E (reverse L)
                                              (setq L* (if (= (setq M (1- M)) N)
                                                           L*
                                                           (cons E L*))))))))
                TOT (+ TOT (* E1 SIG (DET MENOR))))))))

; Per resoldre sistemes del tipus              A1·X1 + B1·X2 + ... + E1·X5 = F1
;                                              A2·X1 + B2·X2 + ... + E2·X5 = F2
;                                              ..............................
;                                              A5·X1 + B5·X2 + ... + E5·X5 = F5
; cal executar SISTEQ, en la forma  (SISTEQ '((A1 B1 ... E1)
```

```
;                                              (A2 B2 ... E2)
;                                              ..............
;                                              (A5 B5 ... E5)) '(F1 F2 ... F5))
; i s'obtindrà com a resultat la llista  (X1 X2 ... X5)
(defun SISTEQ (XXX YY / M N LON DIV XXY XX* SOL)
  (setq LON (length YY)
        DIV (DET XXX) N -1)
  (repeat LON
    (setq N (1+ N)
          XXY (mapcar '(lambda (XX Y)
                         (setq M LON XX* ())
                         (foreach X (reverse XX)
                           (setq XX* (cons (if (= (setq M (1- M)) N) Y X)
                                           XX*))))
                      XXX YY)
          SOL (append SOL (list (/ (DET XXY) DIV))))))

; Per resoldre sistemes del tipus         A·X1^4 + B·X1^3 + ... + D·X1 + E = Y1
;                                         A·X2^4 + B·X2^3 + ... + D·X2 + E = Y2
;                                         ....................................
;                                         A·X5^4 + B·X5^3 + ... + D·X5 + E = Y5
; cal executar AXN, en la forma  (AXN '(X1 X2 ... X5) '(Y1 Y2 ... Y5))
; i s'obtindrà com a resultat la llista  (A B ... D E)
(defun AXN (XX YY / L M N X Y ZZ XXX)
  (if (setq L (member 0 XX))
      (setq N (- (length XX) (length L))
            XX (append (reverse (cdr (member 0 (reverse XX)))) (cdr L))
            Y (nth N YY) M -1
            YY (foreach E YY
                 (setq ZZ (if (= (setq M (1+ M)) N)
                              ZZ
                              (append ZZ (list (- E Y))))))))
  (foreach E XX
    (setq X (list (if Y E 1))
          XXX (append XXX (list (repeat (1- (length XX))
                                  (setq X (cons (* (car X) E) X)))))))
  (append (SISTEQ XXX YY) (list Y)))
```

Sobre aquesta base muntarem les funcions **C:DEF-TRANSF** i **APL-TRANSF** que, com el seu nom indica ens serviran per obtenir la matriu de transformació dels punts, prèvia la introducció de les dues quaternes de punts definidors (els originals i els seus transformats), en un dispositiu similar al de la funció **INSERTOK** de ***INS2D***/***INS3D***, i per aplicar-la als punts que l'usuari vagi introduint, amb una alternativa quant al tipus de sortida: gràfica, dibuixant línies virtuals que es volatilitzaran amb un simple redibuixat (**C:APL-TRANSF-V**); numèrica, representant cada parell de punts (primitiu i transformat) com a llistes de coordenades (**C:APL-TRANSF-N**). Abans, com que **SISTEQ** és una funció genèrica en el sentit que està concebuda per resoldre sistemes de **n** equacions lineals amb **n** incògnites mitjançant la regla de Cramer, mirarem de simplificar-la, ajustant-la a sistemes de quatre: n'eliminarem **XXX** (argument que no necessitem, perquè la funció sempre s'invoca des de **C:DEF-TRANSF**, amb la quaterna **4P** de punts originals), les variables **LON** (**4**), **DIV** (el determinant de **4P**, que passa a ser variable de **C:DEF-TRANSF** perquè l'haurem d'utilitzar abans, i suprimirem les línies 2 i 3 (tret de la inicialització de **N**). En ser recursiva, la funció **DET** no la podrem simplificar de forma anàloga.

```
(defun SISTEQ (YY / M N XXY XX* SOL)
  (setq N -1)
  (repeat 4
    (setq N (1+ N)
          XXY (mapcar '(lambda (XX Y)
                         (setq M 4 XX* ())
                         (foreach X (reverse XX)
                           (setq XX* (cons (if (= (setq M (1- M)) N) Y X)
                                           XX*))))
                      4P YY)
          SOL (append SOL (list (/ (DET XXY) DIV))))))
```

```
(defun LDEP (D PP / VV V1 V2 V3 V4)
  (setq VV (mapcar '(lambda (P) (reverse (cdr (reverse P)))) PP)
        V1 (car VV) V2 (cadr VV) V3 (caddr VV) V4 (last VV))
  (equal (/ D (* (distance V1 V4) (distance V2 V4) (distance V3 V4)))
         0 (expt Q0 2)))

(defun C:DEF-TRANSF (/ Q0 U N P P1 P2 P3 P4 4P P* P*1 P*2 P*3 P*4 4P* X Y Z DIV)
  (setq Q0 0.001 U 2)
  (while (> U 0)
    (foreach N '("1" "2" "3" "4")
      (if (> U 1)
          (progn
            (setq P (getpoint (strcat "\n\nPunto n° " N ": ")))
            (set (read (strcat "P" N)) (reverse (cons 1 (reverse P))))))
      (setq P* (getpoint (if (> U 1)
                             P
                             (reverse
                               (cdr (reverse (eval (read (strcat "P" N)))))))
                         (strcat "\nTransformado del punto n° " N ": ")))
      (set (read (strcat "P*" N)) (reverse (cons 1 (reverse P*)))))
    (setq 4P (list P1 P2 P3 P4)
          4P* (list P*1 P*2 P*3 P*4)
          DIV (DET 4P) N (DET 4P*)
          U (if (LDEP DIV 4P)
                2
                (if (LDEP N 4P*) 1 0)))
    (if (> U 0)
        (alert (strcat "Los cuatro puntos " (if (> U 1) "" "transformados ")
                       "son linealmente dependientes\n(son coplanarios, tres están "
                       "alineados o dos coinciden):\nhay que introducirlos de nuevo"
                         (if (> U 1) ", con sus transformados." "."))))))
  (setq X (SISTEQ (mapcar 'car 4P*))
        Y (SISTEQ (mapcar 'cadr 4P*))
        Z (SISTEQ (mapcar 'caddr 4P*))
        U (list X Y Z '(0 0 0 1))
        U (list (mapcar 'car U) (mapcar 'cadr U) (mapcar 'caddr U)
                (mapcar 'last U))
        **U** U)
  (princ))

(defun C:APL-TRANSF-V () (APL-TRANSF T))

(defun C:APL-TRANSF-N () (APL-TRANSF ()))

(defun APL-TRANSF (VECS / P P* Q Q*)
  (while (progn
           (prompt "\nPunto a transformar: ")
           (setq Q (getpoint (if (and VECS P) P ""))))
         (setq Q* (reverse (cdr (reverse (car (AB (append Q '(1)) **U**))))))
         (if VECS
             (progn
               (if P
                   (progn
                     (grvecs (list 1 P Q))
                     (grvecs (list 3 P* Q*))))
               (setq P Q P* Q*))
             (progn
               (terpri) (princ Q) (princ " -> ") (princ Q*))))
  (princ))
```

Igual com passava a la funció **INSERTOK** de ***INS2D***/***INS3D***, no ens molestem a detectar en temps real la introducció d'un punt improcedent per part de l'usuari, cursant de seguida l'avís per evitar-li feina inútil, sinó que esperem que els hagi entrat tots per informar-lo de la necessitat de tornar a començar. Però llavors almenys l'avís era específic, i podia saber si hi havia 3 punts alineats o 4 de coplanaris (només la repetició d'algun dels tres primers punts donava lloc a la interrupció immediata del procés, per *dividir por cero*), mentre que ara ens limitem a dir-li que els punts introduïts (s'entén que parlem de les coordenades) són linealment

dependents. La decisió quedava justificada per la comoditat d'aprofitar el càlcul d'un determinant **4 × 4** (**DIV** el necessitem en els tres accessos a **SISTEQ**, i només haurem de calcular expressament **N**, corresponent als punts transformats), perquè el determinant dels punts **P1**, **P2**, **P3** i **P4**, representats en coordenades homogènies, té el mateix valor que el determinant **3 × 3** dels vectors $\mathbf{P_1P_2}$, $\mathbf{P_1P_3}$ i $\mathbf{P_1P_4}$, que és precisament el producte mixt $(\mathbf{P_1P_2} \times \mathbf{P_1P_3}) \cdot \mathbf{P_1P_3}$. Però el problema es presenta a l'hora d'adoptar un marge de tolerància per establir quan cal considerar nuls els determinants, perquè si allò que comparem amb la tolerància **Q0** és la relació entre el volum del paral.lelepípede definit pels tres vectors (que, com sabrà el lector, coincideix amb el seu producte mixt) i el del paral·lelepípede ortogonal definit pels seus mòduls, ¿haurem de comparar-la amb **Q0**, amb el seu quadrat (com hem fet a la funció **LDEP**) o amb el seu cub? Doncs dependrà del significat que li donem a **Q0** (tolerància lineal, superficial o volumètrica), però si li mantenim el paper de tolerància angular assignat a ***INS2D***/***INS3D*** no queda gens clar que puguem plantejar-ho així sinó que, tot aprofitant el determinant **DIV**, haurem de fer la comprovació en dues etapes, com en **INSERTOK**. I ja que a l'últim capítol no ens havíem prodigat en explicacions a l'hora de fer-ne l'anàlisi i proposar un dispositiu, ara ens hi esplaiarem una mica més, raó per la qual convindrà superar l'ambigüitat pròpia de la notació que utilitzàvem, en què $\mathbf{P_mP_n}$ tant podia representar un vector com la distància entre $\mathbf{P_m}$ i $\mathbf{P_n}$ (el seu mòdul), i el significat correcte havia d'inferir-se pel context: a partir d'ara adoptarem la de l'article reproduït, on $\mathbf{P_mP_n}$ només respon a la segona accepció i un vector ha de representar-se com $\overline{\mathbf{P_m\ P_n}}$; pel que fa al mòdul de $\overline{\mathbf{P_m\ P_n}}$, no usarem la notació $|\overline{\mathbf{P_m\ P_n}}|$ sinó que ens permetrem la llicència de representar-lo com a distància $\mathbf{OP_m\ P_n}$, on **O** seria l'origen de coordenades.

Si anomenem α l'angle $\mathbf{P_2P_1P_3}$ i β el que forma $\overline{\mathbf{P_1\ P_4}}$ amb el pla de l'anterior, el mòdul del producte vectorial $\mathbf{P_1\ P_2} \times \mathbf{P_1\ P_3}$ és l'àrea del paral·lelogram definit pels dos vectors, és a dir:

$$\mathbf{O}-\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) = \mathbf{P_1P_2} \cdot \mathbf{P_1P_3} \cdot \sin\alpha \quad \rightarrow \quad \sin\alpha = \frac{\mathbf{O}-\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right)}{\mathbf{P_1P_2} \cdot \mathbf{P_1P_3}} > \mathbf{Q0} \quad \rightarrow \quad \alpha > \mathbf{Q0}$$

Evidentment, l'assimilació $\sin\alpha \approx \alpha$ sols és vàlida per a petits angles expressats en radiants. Si ara considerem que el producte mixt $\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) \cdot \overline{\mathbf{P_1\ P_4}}$, en valor absolut, és el volum del paral·lelepípede definit pels tres vectors:

$$\left|\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) \cdot \overline{\mathbf{P_1\ P_4}}\right| = \mathbf{O}-\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) \cdot \mathbf{P_1P_4} \cdot \cos(90°-\beta) = \mathbf{O}-\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) \cdot \mathbf{P_1P_4} \cdot \sin\beta$$

$$\sin\beta = \frac{\left|\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) \cdot \overline{\mathbf{P_1\ P_4}}\right|}{\mathbf{O}-\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) \cdot \mathbf{P_1P_4}} > \mathbf{Q0} \quad \rightarrow \quad \beta > \mathbf{Q0},$$ amb les mateixes reserves que α.

Si ara substituïm la primera equivalència en el denominador, tindrem:

$$\sin\beta = \frac{\left|\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) \cdot \overline{\mathbf{P_1\ P_4}}\right|}{\mathbf{P_1P_2} \cdot \mathbf{P_1P_3} \cdot \mathbf{P_1P_4} \cdot \sin\alpha} \quad \rightarrow \quad \frac{\left|\left(\overline{\mathbf{P_1\ P_2}} \times \overline{\mathbf{P_1\ P_3}}\right) \cdot \overline{\mathbf{P_1\ P_4}}\right|}{\mathbf{P_1P_2} \cdot \mathbf{P_1P_3} \cdot \mathbf{P_1P_4}} = \sin\alpha \cdot \sin\beta > \mathbf{Q0}^2$$

Però atenció!: que $\sin\alpha \cdot \sin\beta \le \mathbf{Q0}^2$ implica que $\sin\alpha \le \mathbf{Q0}$, $\sin\beta \le \mathbf{Q0}$ o que fins i tot s'acompleixin les dues desigualtats alhora, però la recíproca no és certa, és a dir que pot ser molt bé que $\sin\alpha \le \mathbf{Q0}$ o que $\sin\beta \le \mathbf{Q0}$ però que malgrat tot sigui $\sin\alpha \cdot \sin\beta > \mathbf{Q0}^2$, raó per la qual refusarem la versió precedent i seguirem, *mutatis mutandis*, l'algorisme adoptat en **INSERTOK**. Els canvis només afecten **LDEP** i **C:DEF-TRANSF**:

```
(defun LDEP (D VV TX / O A B C A*B I J K)
  (setq O '(0 0 0)
        A (mapcar '- (cadr VV) (car VV))
        B (mapcar '- (caddr VV) (car VV))
        C (mapcar '- (last VV) (car VV))
        A*B (list (- (* (cadr A) (last B)) (* (last A) (cadr B)))
                  (- (* (last A) (car  B)) (* (car  A) (last B)))
                  (- (* (car  A) (cadr B)) (* (cadr A) (car  B))))
        I (distance O A*B)
        J (/ I (* (distance O A) (distance O B)))
        K (/ D (* I (distance O C)))
        TXT (if (<= J Q0)
                "3 primeros"
                (if (equal K 0 Q0) "4"))
        TXT (if TXT (strcat "\nLos " TXT " puntos " TX
                      (if (= TXT "4") "son coplanarios." "están alineados."))))))
```

```
(defun C:DEF-TRANSF (/ Q0 U N P PP P1 P2 P3 P4 4P P* P*P* P*1 P*2 P*3 P*4 4P*
                       XYZ DIV TXT)
  (setq Q0 0.001 U 2)
  (while (> U 0)
    (setq PP () P*P* ())
    (foreach N '("1" "2" "3" "4")
      (if (> U 1)
          (progn
            (setq P (getpoint (strcat "\n\nPunto n° " N ": "))
                  PP (append PP (list P)))
            (set (read (strcat "P" N)) (append P '(1)))))
      (setq P* (getpoint (if (> U 1)
                             P
                             (reverse
                               (cdr (reverse (eval (read (strcat "P" N)))))))
                         (strcat "\nTransformado del punto n° " N ": "))
            P*P* (append P*P* (list P*)))
      (set (read (strcat "P*" N)) (append P* '(1))))
    (setq 4P (list P1 P2 P3 P4) 4P* (list P*1 P*2 P*3 P*4)
          DIV (if PP (DET 4P) DIV)
          N (if P*P* (DET 4P*) N)
          U (if (and PP (LDEP DIV PP ""))
                2
                (if (and P*P* (LDEP N P*P* "transformados ")) 1 0)))
    (if (> U 0)
        (alert (strcat TXT "\nHay que introducirlos de nuevo"
                       (if (> U 1) ", con sus transformados." "."))))))
  (setq X (SISTEQ (mapcar 'car 4P*))
        Y (SISTEQ (mapcar 'cadr 4P*))
        Z (SISTEQ (mapcar 'caddr 4P*))
        U (list X Y Z '(0 0 0 1))
        U (list (mapcar 'car U) (mapcar 'cadr U) (mapcar 'caddr U)
                (mapcar 'last U))
        **U** U)
  (princ))
```

Amb això hem resolt el cas més general: determinació de les transformacions afins per **4** parells de punts linealment independents, en què haurem d'integrar també les isomètriques que incloguin simetria (és el preu que haurà de pagar l'usuari per tenir-ho còmode quan no se n'inclogui, podent-la determinar amb **3** parells de punts sense necessitat que dos d'ells representin posicions homòlogues). Queden, doncs, els altres tres casos: únicament un (**3** parells de punts) compartirà amb el resolt la utilització de **SISTEQ** (com s'explica a l'article, el quart parell de punts el posa "d'ofici" el sistema, per considerar més rendible aprofitar aquest dispositiu general que no pas muntar-ne un altre d'específic; pel que fa als dos restants, sí que surt a compte de tractar-los separadament.

L'ampliació del programa a tot el repertori comportarà la presentació inicial del menú d'opcions corresponent, requisit que la funció **INTRO** resoldrà expeditivament en l'àrea de text, sense complicar-se la vida amb finestres d'edició controlades per arxius .DCL. Per contra, sí que ens molestarem a prevenir qualsevol incidència a l'entrada de dades: així com en ***INS2D***/***INS3D***, que hem anat desenvolupant al llarg de la segona part, només havíem d'introduir els punts transformats (dels primitius només se'ns demanava la distància al punt base, i això calia haver-ho mirat abans) ara l'usuari haurà d'anar alternant originals i transformats, cosa que augmenta el risc de confondre's i repetir una posició ja introduïda; i com que això originaria una enutjosa interrupció per error (divisió per zero) abans que apareguéssin els missatges *Los 3 primeros puntos estan alineados* o bé *Los 4 puntos son coplanarios*, evitarem aquesta contingència amb una funció **GETPUNT** interposada a **getpoint**.

Atès el propòsit complementari del present annex respecte al gruix del treball, no entrarem en detalls sobre artificis de programació (els aclariments indispensables s'han incorporat al codi com a comentaris que precedeixen l'expressió avaluable), i només farem una excepció en allò relatiu a la modalitat **B** de la definició per **3** punts, encara que només sigui perquè en realitat no figurava a la versió original de l'article i a la d'ara només n'hem fet una menció fugissera, com qui no vol la cosa, a l'apartat 5.- DEFINICIÓ PER 1 PARELL DE PUNTS i en el quadre sinòptic que el seguia. En particular, convindria aclarir perquè en l'opció **B** (translació, gir

i escalat uniforme) utilitzem la mateixa expressió **[12]** de l'opció isomètrica **A** (exclusivament translació i gir) per calcular P'_4 (un P'_4 diferent en ambdós casos, perquè P'_2 també ho és) i per calcular P'_3 modifiquem el resultat obtingut en **[10]**.

Per fer-ho, retornarem a l'apartat 3.- DEFINICIÓ PER 3 PARELLS DE PUNTS, per bé que canviant la notació de P'_2, P'_3 i P'_4 per P'_{2a}, P'_{3a} i P'_{4a}, per diferenciar aquests punts dels fins ara homònims en l'opció **B**, que passarem a anomenar P'_{2b}, P'_{3b} i P'_{4b}.

Així, en la variant **A** fèiem $\overline{P'_1\,P'_{2a}} = \overline{P'_1\,Q'}\,\frac{P_1-P_2}{P'_1-Q'}$ **[6]** per tal que fos $P'_1-P'_{2a} = P_1-P_2$, i calculàvem $\overline{U} = \overline{P_1\,P_2} \times \overline{P_1\,R} = \overline{P_1\,P_3}$ **[7]** i $\overline{V'_a} = \overline{P'_1\,P'_{2a}} \times \overline{P'_1\,S'}$ **[8]**. Adoptàvem $\overline{P_1\,P_3} = \overline{U}$ **[9]**, però com que havia de ser $P'_1-P'_{3a} = P_1-P_3$ fèiem $\overline{P'_1\,P'_{3a}} = \overline{V'_a}\,\frac{P_1-P_3}{O-V'_a} = \overline{V'_a}\,\frac{O-U}{O-V'_a}$ **[10]**. Com que $\overline{P_1\,P_3}$ era perpendicular a $\overline{P_1\,P_2}$ i $\overline{P'_1\,P'_{3a}}$ ho era a $\overline{P'_1\,P'_{2a}}$, $\overline{P_1\,P_4} = \overline{P_1\,P_2} \times \overline{P_1\,P_3}$ **[11]** era perpendicular a $\overline{P_1\,P_2}$ i $\overline{P_1\,P_3}$, i $\overline{W'_a} = \overline{P'_1\,P'_{2a}} \times \overline{P'_1\,P'_{3a}}$ **[12]** ho era a $\overline{P'_1\,P'_{2a}}$ i $\overline{P'_1\,P'_{3a}}$, però no només això sinó que $O-W'_a = P'_1-P'_2 \cdot P'_1-P'_3 = P_1-P_2 \cdot P_1-P_3 = P_1-P_4$, raó per la qual adoptàvem directament $\overline{P'_1\,P'_{4a}} = \overline{W'_a}$ (just ens interessava que $P'_1-P'_{4a} = P_1-P_4$). En la variant **B** serà $P'_1-P'_{2b} = P'_1-Q' \neq P_1-P_2$, i anomenarem $E = \frac{P'_1-P'_{2b}}{P_1-P_2} = \frac{P'_1-Q'}{P_1-P_2} \neq 1$, és a dir que $P'_1-P'_{2b} = E \cdot P_1-P_2 = E \cdot P'_1-P'_{2a}$. Com en **A**, calcularem $\overline{U} = \overline{P_1\,P_2} \times \overline{P_1\,R} = \overline{P_1\,P_3}$ i un $\overline{V'_b} = \overline{P'_1\,P'_{2b}} \times \overline{P'_1\,S'}$ orientat com $\overline{V'_a}$ (P'_{2a} i P'_{2b} estan alineats amb P'_1) però amb un mòdul diferent (perquè $P'_1-P'_{2b} = E \cdot P_1-P_2$), i com que ens interessa que $P'_1-P'_{3b} = E \cdot P_1-P_3 = E \cdot P'_1-P'_{3a}$, prendrem $\overline{P'_1\,P'_{3b}} = \overline{V'_b}\,E\,\frac{P_1-P_3}{O-V'_b} = \overline{V'_b}\,E\,\frac{O-U}{O-V'_b} = \overline{V'_a}\,E\,\frac{O-U}{O-V'_a} = E \cdot \overline{P'_1\,P'_{3a}}$.

Com que, anàlogament a **A**, $\overline{P_1\,P_3}$ és perpendicular a $\overline{P_1\,P_2}$ i $\overline{P'_1\,P'_{3a}}$ ho és a $\overline{P'_1\,P'_{2a}}$, si fem $\overline{P_1\,P_4} = \overline{P_1\,P_2} \times \overline{P_1\,P_3}$ i $\overline{W'_b} = \overline{P'_1\,P'_{2b}} \times \overline{P'_1\,P'_{3b}}$ resultarà que $\overline{P_1\,P_4}$ és perpendicular a $\overline{P_1\,P_2}$ i $\overline{P_1\,P_3}$, i que $\overline{W'_b}$ ho és a $\overline{P'_1\,P'_{2b}}$ i $\overline{P'_1\,P'_{3b}}$, però no prendrem $\overline{P'_1\,P'_{4b}} = \overline{W'_b}$ perquè $O-W'_b = P'_1-P'_{2b} \cdot P'_1-P'_{3b} = E^2 \cdot P_1-P_2 \cdot P_1-P_3 = E^2 \cdot P_1-P_4$ ($= E^2 \cdot P'_1-P'_{2a} \cdot P'_1-P'_{3a} = E^2 \cdot O-W'_a$) i ens interessa que $P'_1-P'_{4b} = E \cdot P_1-P_4$, així que farem $\overline{P'_1\,P'_{4b}} = \frac{\overline{W'_b}}{E} = E \cdot \overline{W'_a} = E \cdot \overline{P'_1\,P'_{4a}}$. Dit això, **C:DEF-TRANSF** i les funcions auxiliars **->P*2**, **->P*3** i **->P*4** no aniran a buscar directament les fórmules específiques sinó que es regiran pels criteris de prioritzar la troncalitat del procés (bifurcar poc) i l'economia de càlcul: una vegada tinguem $P'_{2b} = Q'$, en comptes d'obtenir P'_{3b} i P'_{4b} basant-nos en les relacions $\overline{P'_1\,P'_{3b}} = \overline{V'_b}\,E\,\frac{P_1-P_3}{O-V'_b}$ ($\overline{V'_b} = \overline{P'_1\,P'_{2b}} \times \overline{P'_1\,S'}$) i $\overline{P'_1\,P'_{4b}} = \frac{\overline{W'_b}}{E}$ ($\overline{W'_b} = \overline{P'_1\,P'_{2b}} \times \overline{P'_1\,P'_{3b}}$), deixarem per al final el producte per **E** i així ens estalviarem una divisió per **E**:

- Començarem fent $\overline{P'_1\,P'_{3c}} = \overline{V'_b}\,\frac{O-U}{O-V'_b} = \frac{\overline{P'_1\,P'_{3b}}}{E} = \overline{P'_1\,P'_{3a}}$ (resultat definitiu en opció **A**).
- Seguirem fent $\overline{W'_c} = \overline{P'_1\,P'_{2b}} \times \overline{P'_1\,P'_{3c}} = \frac{\overline{W'_b}}{E} = \overline{P'_1\,P'_{4b}}$ (procediment comú a opcions **A** i **B**).
- Acabarem fent $\overline{P'_1\,P'_{3b}} = E \cdot \overline{P'_1\,P'_{3c}}$ (només en opció **B**).

Aclarit això, presentem el codi sencer, tant aquell que no ha sofert canvis com el que hem modificat i ampliat, amb les noves funcions auxiliars:

```
; Multiplica la matriu A (NIxNK) per la matriu B (NKxNJ).
; Les matrius han de representar-se com a llistes de subllistes-fila; els vectors,
; tant si són v-fila (A) com v-columna (B), poden introduir-se com a llistes
; simples (la funció ja els formatejarà adequadament).
(defun AB (A B / C AxB AIxB AIxBJ NI NJ NK I J K)
  (if (atom (car A)) (setq A (list A)))
  (if (atom (car B)) (progn
                       (foreach E B (setq C (append C (list (list E)))))
                       (setq B C)))
```

```
  (if (= (setq NK (length (car A))) (length B))
    (progn
      (setq NI (length A) NJ (length (car B)) I -1)
      (repeat NI
        (setq I (1+ I) J -1 AIxB ())
        (repeat NJ
          (setq J (1+ J) K -1 AIxBJ 0)
          (repeat NK
            (setq K (1+ K)
                  AIxBJ (+ AIxBJ (* (nth K (nth I A)) (nth J (nth K B))))))
          (setq AIxB (append AIxB (list AIxBJ))))
        (setq AxB (append AxB (list AIxB)))))))

(defun TRANSPOSAR (A / LLE LL LE E LS LLS)
  (setq LLE A)
  (while LLE
         (setq LS () LL LLE)
         (foreach L LL
                  (setq E (car L) LE (cdr L)
                        LS (append LS (list E))
                        LLE (if (not (equal LLE LL)) LLE)
                        LLE (if LE (append LLE (list LE)))))
         (setq LLS (append LLS (list LS)))))

; Adapta el tercer element A de NENTSEL al format del tercer element de NENTSELP,
; requerit pel segon argument (optatiu) de GRVECS. Altres exigències de GRVECS són
; que SCP=SCU i que el primer argument respongui a una inserció neutra a l'origen.
(defun ADAPSEL (A) (append (TRANSPOSAR A) (list '(0 0 0 1))))

(defun DET (LL / M N LON SIG TOT MENOR L*)
  (if (= 2 (setq LON (length LL)))
      (- (* (car (car LL))(cadr (cadr LL)))
         (* (car (cadr LL))(cadr (car LL))))
      (progn
        (setq TOT 0 SIG -1 N -1)
        (foreach E1 (car LL)
          (setq SIG (- SIG) N (1+ N) MENOR ()
                MENOR (foreach L (cdr LL)
                        (setq M LON L* ()
                              MENOR (append MENOR
                                      (list (foreach E (reverse L)
                                              (setq L* (if (= (setq M (1- M)) N)
                                                           L*
                                                           (cons E L*))))))))
                TOT (+ TOT (* E1 SIG (DET MENOR))))))))

(defun SISTEQ (YY / M N XXY XX* SOL)
; Per resoldre sistemes del tipus              A1·X1 + B1·X2 + ... + E1·X5 = F1
;                                              A2·X1 + B2·X2 + ... + E2·X5 = F2
;                                              ..............................
;                                              A5·X1 + B5·X2 + ... + E5·X5 = F5
; cal executar SISTEQ, en la forma  (SISTEQ '((A1 B1 ... E1)
;                                             (A2 B2 ... E2)
;                                             ..............
;                                             (A5 B5 ... E5)) '(F1 F2 ... F5))
; i s'obtindrà com a resultat la llista  (X1 X2 ... X5)
  (setq N -1)
  (repeat 4
    (setq N (1+ N)
          XXY (mapcar '(lambda (XX Y)
                         (setq M 4 XX* ())
                         (foreach X (reverse XX)
                           (setq XX* (cons (if (= (setq M (1- M)) N) Y X)
                                           XX*))))
                      4P YY)
          SOL (append SOL (list (/ (DET XXY) DIV))))))
```

```
(defun AXN (XX YY / L M N X Y ZZ XXX)
; Per resoldre sistemes del tipus         A·X1^4 + B·X1^3 + ... + D·X1 + E = Y1
;                                         A·X2^4 + B·X2^3 + ... + D·X2 + E = Y2
;                                         ...................................
;                                         A·X5^4 + B·X5^3 + ... + D·X5 + E = Y5
; cal executar AXN, en la forma  (AXN '(X1 X2 ... X5) '(Y1 Y2 ... Y5))
; i s'obtindrà com a resultat la llista  (A B ... D E)
  (if (setq L (member 0 XX))
      (setq N (- (length XX) (length L))
            XX (append (reverse (cdr (member 0 (reverse XX)))) (cdr L))
            Y (nth N YY) M -1
            YY (foreach E YY
                 (setq ZZ (if (= (setq M (1+ M)) N)
                              ZZ
                              (append ZZ (list (- E Y))))))))
  (foreach E XX
    (setq X (list (if Y E 1))
          XXX (append XXX (list (repeat (1- (length XX))
                                  (setq X (cons (* (car X) E) X)))))))
  (append (SISTEQ XXX YY) (list Y)))

(defun GETPUNT (P TXT / Q QQ)
  (setq QQ (if (> (strlen TXT) 11) P*P* PP))
  (while (member (setq Q (initget 1)
                       Q (if P (getpoint P TXT) (getpoint TXT))) QQ)
         (prompt "\nPunto repetido."))
  Q)

; PRODUCTE VECTORIAL dels vectors O->A i O->B, on OAB és una llista del tipus
; (list O A B ...), d'almenys tres punts 3D: O A i B. El resultat és un "dotted
; pair" on el segon element és el vector producte, i el primer indica si el
; quocient entre el mòdul d'aquest producte i el producte dels mòduls de O->A
; i de O->B és o no mes gran que Q0.
 (defun 12*13 (OAB / A B AB)
  (setq A (mapcar '- (nth 1 OAB) (nth 0 OAB))
        B (mapcar '- (nth 2 OAB) (nth 0 OAB))
        AB (list (- (* (nth 1 A) (nth 2 B)) (* (nth 2 A) (nth 1 B)))
                 (- (* (nth 2 A) (nth 0 B)) (* (nth 0 A) (nth 2 B)))
                 (- (* (nth 0 A) (nth 1 B)) (* (nth 1 A) (nth 0 B)))))
  (cons (> (distance O AB) (* (distance O A) (distance O B) Q0)) AB))

(defun LDEP (D VV TX / W W>0 C I J)
  (if (= **N** "3")
      (setq W>0 VV)
      (progn
        (setq W (12*13 VV) W>0 (car W))
        (if W>0
            (setq I (distance O (cdr W))
                  C (mapcar '- (last VV) (car VV))
                  J (/ D (* I (distance O C)))))))
  (setq TXT (if W>0
                (if (equal J 0 Q0) "4")
                (if (= **N** "3") "3" "3 primeros"))
        TXT (if TXT (strcat "\nLos " TXT " puntos " TX
                      (if (= TXT "4") "son coplanarios." "están alineados.")))))

(defun INTRO ()
  (textscr)
  (if (not **N**) (setq **N** "4"))
  (prompt "\n\nDefine una transformación lineal en 3D. Puedes hacerlo con:")
  (prompt "\n- 1 par de puntos (traslación)")
  (prompt "\n- 2 pares de puntos (traslación y escalado)")
  (prompt "\n- 3 pares de puntos (traslación y giro, opcionalmente con escalado ")
  (prompt "uniforme\n- 4 pares de puntos (traslación, giro y escalado)")
  (initget "1 2 3 4")
  (setq N (getkword (strcat "\n¿Con cuántos pares de puntos? <" **N** ">: "))
        **N** (if N N **N**))
```

```
  (if (= **N** "3")
      (progn
        (if (not **M**) (setq **M** "A"))
        (initget "A B")
        (setq M (getkword (strcat "\n- Opción A: Conservar la métrica original "
                                  "(usar sólo traslación y giro)"
                                  "\n- Opción B: Modificar la métrica "
                                  "(traslación, giro y escalado uniforme)"
                                  "\nEscoge A o B <" **M** ">: "))
              M (if M M **M**) **M** M)))
  (graphscr))

(defun BLA-BLA ()
  (if (= N "2")
      (strcat "el transformado de 1, define la semirrecta"
              "\nhomóloga de la determinada por los puntos 1 y 2: ")
      (strcat "los transformados precedentes, define el semiplano"
              "\nhomólogo del determinado por los puntos 1, 2 y 3: ")))

(defun ->P*2 ()
  (setq P*1 (reverse (cdr (reverse P*1))))
  (if (= M "A")
      (setq D/D (/ (distance (car PP) (cadr PP)) (distance P*1 P*))
            P*2 (mapcar '(lambda (C*1 C*) (+ C*1 (* D/D (- C* C*1)))) P*1 P*))
      (setq P*2 P*)))

(defun ->P*3 ()
  (setq U (12*13 PP)
        PP (if (car U) PP) U (cdr U)
        V (12*13 P*P*)
        P*P* (car V) V (cdr V)))

(defun ->P*4 ()
  (if PP
      (setq PP* PP P3 (mapcar '+ (car PP) U)
            P4 (mapcar '+ (car PP) (cdr (12*13 (subst P3 (last PP) PP))))
            P3 (append P3 '(1)) P4 (append P4 '(1))))
  (if P*P*
      (setq D/D (/ (distance O U) (distance O V))
            P*3 (mapcar '(lambda (C*1 CV) (+ C*1 (* D/D CV))) P*1 V)
            P*P* (list P*1 P*2 P*3)
            P*4 (mapcar '+ P*1 (cdr (12*13 P*P*)))
            D/D (if (= M "A")
                    D/D
                    (/ (distance P*1 P*2) (distance (car PP) (cadr PP))))
            P*3 (if (= M "A")
                    P*3
                    (mapcar '(lambda (C*1 C*3) (+ C*1 (* D/D (- C*3 C*1))))
                            P*1 P*3))
            P*1 (append P*1 '(1)) P*2 (append P*2 '(1))
            P*3 (append P*3 '(1)) P*4 (append P*4 '(1)))))

(defun AVIS ()
  (alert (strcat "\nTRANSFORMACIÓN CON SIMETRÍA"
                 "\n\nTal vez haya que volver a definirla"
                 "\no invertir la orientación de las caras")))

(defun C:DEF-TRANSF (/ Q0 O K U V M N P PP P1 P2 P3 P4 4P PP* P* P*P* P*1 P*2 P*3
                       P*4 4P* D/D X Y Z DIV TXT)
  (INTRO)
  (setq Q0 0.001 O '(0 0 0)
        K (if (< **N** "2") 0 2))
  (if (< K 1)
      (setq P (GETPUNT () "\n\nPunto original: ")
            P* (GETPUNT P "\n\nPunto transformado: ")
            **K** (list '(1 0 0 0) '(0 1 0 0) '(0 0 1 0)
                        (append (mapcar '- P* P) '(1)))))
```

```
(while (> K 0)
  (setq PP* (if (= **N** "3") PP* PP)
        PP (if (< K 2) PP) P*P* ())
  (foreach N (reverse (member **N** '("4" "3" "2" "1")))
    (if (> K 1)
        (progn
          (setq P (GETPUNT () (strcat "\n\nPunto " N ": "))
                PP (append PP (list P)))
          (set (read (strcat "P" N)) (append P (if (> **N** "2") '(1))))))
    (setq P* (GETPUNT (if (> K 1)
                          P
                          (nth (1- (atoi N)) PP*))
                      (if (and (= **N** "3")
                               (or (= N "3")
                                   (and (= N "2") (= M "A"))))
                          (strcat "\nPunto que, con " (BLA-BLA))
                          (strcat "\nTransformado del punto " N ": ")))
          P*P* (append P*P* (list P*)))
    (set (read (strcat "P*" N)) (append P* (if (> **N** "2") '(1))))
    (if (and (= **N** "3") (> N "1"))
        (if (= N "2") (->P*2) (->P*3))))
  (if (> **N** "2")
      (progn
        (if (= **N** "3") (->P*4))
        (setq 4P (list P1 P2 P3 P4)
              4P* (list P*1 P*2 P*3 P*4)
              DIV (if PP (DET 4P) DIV)
              N (if P*P* (DET 4P*) N)
              K (if (and (or PP (= **N** "3")) (LDEP DIV PP ""))
                    2
                    (if (and (or P*P* (= **N** "3"))
                             (LDEP N P*P* "transformados ")) 1 0))))
      (progn
        (setq U (* (distance P1 P2) Q0)
              M (- (car P2) (car P1))
              N (- (cadr P2) (cadr P1))
              O (- (last P2) (last P1))
              V (* (distance P*1 P*2) Q0)
              X (- (car P*2) (car P*1))
              Y (- (cadr P*2) (cadr P*1))
              Z (- (last P*2) (last P*1)))
        (if (or (and (> (abs M) U) (<= (abs X) V))
                (and (<= (abs M) U) (> (abs X) V))
                (and (> (abs N) U) (<= (abs Y) V))
                (and (<= (abs N) U) (> (abs Y) V))
                (and (> (abs O) U) (<= (abs Z) V))
                (and (<= (abs O) U) (> (abs Z) V)))
            (setq TXT (strcat "\nLos 2 pares de puntos no definen"
                              "\ntraslaciones ni escalados."))
            (setq K 0))))
  (if (> K 0)
      (alert (strcat TXT "\nHay que introducirlos de nuevo"
                     (if (and (> **N** "2") (> K 1))
                         ", con sus transformados."
                         "."))))))
(if (> **N** "2")
    (setq X (SISTEQ (mapcar 'car 4P*))
          Y (SISTEQ (mapcar 'cadr 4P*))
          Z (SISTEQ (mapcar 'caddr 4P*))
          K (list X Y Z '(0 0 0 1))
          K (list (mapcar 'car K) (mapcar 'cadr K)
                  (mapcar 'caddr K) (mapcar 'last K))
          **K** K)
    (if (> **N** "1")
        (setq X (if (and (> (abs M) U) (> (abs X) V)) (/ X M) 1)
              Y (if (and (> (abs N) U) (> (abs Y) V)) (/ Y N) 1)
              Z (if (and (> (abs O) U) (> (abs Z) V)) (/ Z O) 1)
```

```
                **K** (list (list X 0 0 0) (list 0 Y 0 0) (list 0 0 Z 0)
                            (list (if (and (> (abs M) U) (> (abs X) V))
                                      (/ (- (* (car P2) (car P*1))
                                            (* (car P1) (car P*2))) M)
                                      (- (car P*1) (car P1)))
                                  (if (and (> (abs N) U) (> (abs Y) V))
                                      (/ (- (* (cadr P2) (cadr P*1))
                                            (* (cadr P1) (cadr P*2))) N)
                                      (- (cadr P*1) (cadr P1)))
                                  (if (and (> (abs O) U) (> (abs Z) V))
                                      (/ (- (* (last P2) (last P*1))
                                            (* (last P1) (last P*2))) O)
                                      (- (last P*1) (last P1)))
                                  1)))))
  (if (or (and (= **N** "2") (< (* X Y Z) 0))
          (and (> **N** "3")
               ; En usar com a argument de DET (list (cdr (reverse X)) ...) i no
               ; (list (reverse (cdr (reverse X))) ...), DET canvia de signe.
               (> (DET (list (cdr (reverse X)) (cdr (reverse Y))
                             (cdr (reverse Z)))) 0)))
      (AVIS))
  (princ))

(defun C:APL-TRANSF-V () (APL-TRANSF T))

(defun C:APL-TRANSF-N () (APL-TRANSF ()))

(defun APL-TRANSF (VECS / P P* Q Q*)
  (while (progn (prompt "\nPunto a transformar: ")
                (setq Q (getpoint (if (and VECS P) P ""))))
         (setq Q* (reverse (cdr (reverse (car (AB (append Q '(1)) **K**))))))
         (if VECS
             (progn
               (if P (progn (grvecs (list 1 P Q)) (grvecs (list 3 P* Q*))))
               (setq P Q P* Q*))
             (progn (terpri) (princ Q) (princ " -> ") (princ Q*))))
  (princ))
```

La funció **C:APL-TRANSF-V** recorre a línies virtuals per no interferir amb el dibuix i poder-se-les treure del damunt fent ***REDIBUJA***, ***REGEN*** o ***REGENT***, perquè ben segur que el codi presentat només estarà a l'abast de qui ja sàpiga què té entre mans. Però fins i tot per aquest tipus d'usuari, acostumat a fer proves i a anul·lar-ne els resultats fent ***DESHACER*** o ***H***, pot resultat molest que aquesta eina habitual no serveixi de res per desempallegar-se de unes línies de estatus tan precari i que, tanmateix, qualsevol ***ZOOM*** o ***ENCUADRE*** ens les engegui a dida. Si aquestes respostes desconcertants haguessin de constituir un problema, suggerim una variant a base de línies de debò (objectes *LINEA*, volem dir), que l'usuari podrà manipular de manera més convencional:

```
(defun APL-TRANSF (VECS / P P* Q Q* COL ECO)
  (if VECS (progn
             (setq COL (getvar "CECOLOR") ECO (getvar "CMDECHO"))
             (command "DESHACER"
                      (progn (if (= ECO 1) (repeat 100 (princ "\10 \10"))) "I"))
             (if (= ECO 1) (progn (princ "\r") (repeat 100 (princ " "))
                                  (princ "\r") (princ)))
             (setvar "CMDECHO" 0)))
  (while (progn (prompt "\nPunto a transformar: ")
                (setq Q (getpoint (if (and VECS P) P ""))))
         (setq Q* (reverse (cdr (reverse (car (AB (append Q '(1)) **K**))))))
         (if VECS
             (progn (if P (command "COLOR" 1 "LINEA" P Q ""
                                   "COLOR" 3 "LINEA" P* Q* ""))
                    (setq P Q P* Q*))
             (progn (terpri) (princ Q) (princ " -> ") (princ Q*))))
  (if VECS (progn (command "COLOR" COL "DESHACER" "F")
                  (setvar "CMDECHO" ECO)))
  (princ))
```

Una altra cosa que a més d'un l'haurà pogut estranyar és com, utilitzant la funció **GRVECS**, no hem recorregut al segon argument opcional (llista de 4 sublllistes-fila, de 4 elements reals cadascuna) que representa la matriu ****GS**** d'una transformació a aplicar als punts del primer argument **VECS**. Considerant que ****GS**** és justament la transposada de ****K****, tindria sentit objectar perquè, en lloc de calcular cada punt **Q*** transformat de **Q**, fent

```
(while (progn (prompt "\nPunto a transformar: ")
              (setq Q (getpoint (if P P ""))))
       (setq Q* (reverse (cdr (reverse (car (AB (append Q '(1)) **K**))))))
       (if P (progn (grvecs (list 1 P Q)) (grvecs (list 3 P* Q*))))
       (setq P Q P* Q*))
```

no podíem haver-nos limitat a calcular un sol cop

```
(setq **GS** (TRANSPOSAR **K**))
```

i deixar el bucle en la forma

```
(while (progn (prompt "\nPunto a transformar: ")
              (setq Q (getpoint (if P P ""))))
       (if P (progn (grvecs (list 1 P Q)) (grvecs (list 3 P Q) **GS**)))
       (setq P Q))
```

Ben segur que l'omissió de les expressions associades a **(not VECS)** us haurà posat sobre la pista, donant-vos la resposta: la conveniència de tenir també una sortida numèrica **C:APL-TRANSF-N**, complementària de la gràfica **C:APL-TRANSF-V**, era la raó més important d'haver-nos molestat a calcular **Q*** (**P***, **Q*** i tots els transformats); però no l'única, perquè amb aquest segon argument els vectors virtuals de **GRVECS** només proporcionen una imatge correcta dels punts transformats si mirem el dibuix en planta (direcció de visualització segons l'eix ***Z*** del ***SCP***), i això és una gaita.

Però aquesta disquisició ens dóna peu a criticar els migrats recursos d'AutoLISP a l'hora de proporcionar-nos aquest argument. Ens referim al tercer element de les llistes que resulten de seleccionar amb les funcions **NENTSEL** i **NENTSELP** insercions de bloc, que representa la transformació que permet situar punts de la definició del bloc (referits al sistema propi d'aquest, paral·lel al ***SCP*** vigent en executar ***BLOQUE*** i amb l'origen en el punt base) sobre la inserció designada (referint-los al ***SCU****niversal*):

- A **NENTSEL** és una matriu **4 × 3**, en què la submatriu **3 × 3** superior és el producte d'un escalat en ***X***, ***Y*** i ***Z*** per un gir al voltant de l'eix ***Z*** i per la conversió del **SCP** al ***SCU****niversal* (considerada en aquesta transformació només l'orientació dels dos sistemes, com si en la funció **TRANS** s'explicités un quart argument no nul), i la quarta fila és el producte del punt d'inserció per l'esmentada conversió ***SCP*** → ***SCU*** (però considerats ara els orígens dels dos sistemes). Punt d'inserció, factors d'escala i angle de gir són, és clar, els de la inserció seleccionada.
- A **NENTSELP** és la matriu transposada de la precedent i homologada a coordenades homogènies amb l'afegitó d'una quarta fila **0 0 0 1**. És doncs una matriu **4 × 4**.

Abans de seguir, recuperem l'alè i comparem el perfil de les quatre matrius a què fins ara hem fet esment:

- El format de ****K****, resultant de **DEF-TRANSF**, és
```
(list (list Ax Ay Az 0)
      (list Bx By Bz 0)
      (list Cx Cy Cz 0)
      (list Dx Dy Dz 1))
```
- El format del tercer element de la llista obtinguda seleccionant amb **NENTSEL** una inserció de bloc, fent **(caddr (nentsel "\nDesigna una inserción: "))**, és
```
(list (list Ax Ay Az)
      (list Bx By Bz)
      (list Cx Cy Cz)
      (list Dx Dy Dz))
```
- El format del tercer element de la llista obtinguda seleccionant amb **NENTSELP** una inserció de bloc, fent **(caddr (nentselp "\nDesigna una inserción: "))**, és
```
(list (list Ax Bx Cx Dx)
      (list Ay By Cy Dy)
      (list Az Bz Cz Dz)
      (list 0  0  0  1))
```
- El format que cal donar al segon argument ****GS**** de la funció **GRVECS** perquè, en visualitzar el dibuix en planta, els vectors virtuals no responguin al punts del primer argument sinó als obtinguts de la transformació que representa ****GS****, és
```
(list (list Ax Bx Cx Dx)
      (list Ay By Cy Dy)
      (list Az Bz Cz Dz)
      (list 0  0  0  1))
```

A primer cop d'ull sembla que no hi hagi d'haver massa problema per obtenir ****GS**** a partir de **NENTSEL**, fent

(setq **GS (ADAPSEL (caddr (nentsel "\nDesigna una inserción: "))))**

o, millor encara, prenent-lo directament de **NENTSELP**, amb

(setq **GS (caddr (nentselp "\nDesigna una inserción: ")))**

però cal tenir present que, malgrat haver usat els mateixos símbols en tots quatre casos, el valor d'elements homònims de la 2ª i 3ª matriu pot no coincidir amb els de la 1ª i 3ª. El problema ve de la disparitat de plantejaments: en el 1r i 4t cas la transformació es defineix entre figures que coexisteixen a l'espai (posicions originals i transformades estan referides al mateix ***SCP***), mentre que en el 2n i 3r cas (a banda que la transformació funciona en el ***SCU****niversal* i que, perquè ho faci en qualsevol altre ***SCP*** s'hagi d'aplicar la conversió ***SCU*** → ***SCP***) tenim la inserció de bloc seleccionada des de **NENTSEL** o **NENTSELP** però ¿on representa que és el bloc? El tenim descrit a la taula de blocs de la base de dades del dibuix, però el bloc *stricto sensu* no es deixa veure: allò que visualitzem són les seves insercions. Encara que conservem la matèria primera utilitzada per crear el bloc (en forma d'objectes independents o d'inserció sense escalar ni girar), la materialització gràfica més fidel seria una inserció just a l'origen del ***SCP*** actual (també sense escalar ni girar), així que, si el bloc no va ser creat de manera que el punt base coincidís amb l'origen de coordenades i ara utilitzem aquesta matèria primera com a figura original, treient-ne punts per al primer argument de **GRVECS**, la matriu obtinguda de **NENTSELP** no funcionarà com a segon argument: pel cap baix, suposant que en tot el procés (definició del bloc i insercions) no ens haguéssim mogut del ***SCU***, aquesta tercera matriu i la quarta (la que necessitaríem de segon argument) diferiran en els valors **Dx**, **Dy** i **Dz**.

Però encara que tinguéssim cura de no sortir del ***SCU*** per no complicar les coses, i que ens molestéssim a realitzar una inserció neutra a l'origen, utilitzant-la per subministrar a **GRVECS** les posicions del primer argument, amb l'esperança de veure com, usant de segon la matriu creada per **NENTSELP** en seleccionar l'altra inserció, les grafiava adaptades a la nova geometria, ens adonaríem d'una severa limitació comú a **NENTSEL** i **NENTSELP**: només serveixen per a insercions simples, perquè quan les fem dobles o triples per aconseguir un efecte de cisallament, és a dir, quan jugant amb blocs en resulti una transformació afí on les alineacions ortogonals del bloc original s'hagin convertit en unes que no formen angle recte (com passa amb ***INS3D*** quan el cub de referència ha d'encabir-se en un paral·lelepípede oblic), d'aquelles que ***DEF-TRANSF*** només pot definir amb l'opció **4**, no hi haurà res a fer. Però ja hem vist com ***DEF-TRANSF*** pot suplir les carències d'aquestes dues funcions. Ves per on, ***DEF-TRANSF*** ja tindria una aplicació pràctica en AutoCAD, si més no: proporcionar-nos un mitjà senzill per obtenir l'argument opcional de **CRVECS**, tot permetent-nos prescindir del galimaties de les funcions **NENTSEL** i **NENTSELP**.